②

ETL-0522

Research in Knowledge-Based Vision Techniques for the Autonomous Land Vehicle Program
Third Annual Report

Editors:	Contributions by:
R. Nevatia	W. Franzen
K. Price	S. Gazit
	G. Medioni
	I. Pavlin
	S. Peng

Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, California 90089-0273

December 1988

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

U.S. Army Corps of Engineers
Engineer Topographic Laboratories
Fort Belvoir, Virginia 22060-5546

IRIS#248

Research in Knowledge-Based Vision Techniques for the Autonomous Land Vehicle Program

Contract #DACA76-85-C-0009

Third Annual Technical Report
covering the period of June 1, 1987 to May 31, 1988

R. Nevatia
(Principal Investigator)
and
K. Price
(Editors)

includes contributions by:
W. Franzen
S. Gazit
G. Medioni
I. Pavlin
S. Peng

Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, CA 90089-0273

December, 1988

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY None 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE None	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) IRIS Report 248	5. MONITORING ORGANIZATION REPORT NUMBER(S) ETL-0522
6a. NAME OF PERFORMING ORGANIZATION University of Southern California 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U.S. Army Engineer Topographic Laboratories
6c. ADDRESS (City, State, and ZIP Code) Institute for Robotics & Intelligent Systems Powell Hall, Room 204 Los Angeles, Ca 90089-0273	7b. ADDRESS (City, State, and ZIP Code) Code: CEETL-RI Fort Belvoir, Va 22060-5546
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Defense Advanced Research Projects Agency 8b. OFFICE SYMBOL (If applicable) DARPA-ISTO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DACA76-85-C-0009
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, Va 22209-2308	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Research in Knowledge-Based Vision Techniques for the Autonomous Land Vehicle Program (Unclassified) Third Annual Technical Report

12. PERSONAL AUTHOR(S)
R. Nevatia and K. Price

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Annual Technical Report	FROM 6/1/87 TO 5/31/88	1988, December	85

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
17	7	
17	8	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Autonomous Land Vehicle, Motion Analysis, Target Detection ~~and Description~~, Knowledge-Based Vision. (ALV)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report describes our research in motion analysis and estimation techniques for the period of June 1, 1987 to May 31, 1988. This research is of particular relevance to the DARPA Autonomous Land Vehicle (ALV) program, but should also be of other general utility.

Our basic approach to detecting and tracking motion is to extract and match features, such as lines and regions, from a sequence and to generate motion estimates from these.

We present one report on matching edge elements in connected line segments (contours) in a sequence of views. This work assumes relatively small motions between views.

We also present a report on an alternative representation for motion and a technique to use occlusion in spatio-temportal analysis.

We also present results from a basic integrated system that combines feature extraction, matching and motion estimation. [illegible]

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Rosalene M. Holecheck	22b. TELEPHONE (Include Area Code) (202) 355-2700	22c. OFFICE SYMBOL CEETL-RI

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Contents

Contents

Contents

List of Figures

List of Tables

1 INTRODUCTION AND SUMMARY

This report describes our research activities on Contract DACA76-85 C-0009 for the period June 1, 1987 through May 31, 1988. The contract title "Knowledge-Based Vision Techniques" is part of the DARPA Strategic Computing Program and is monitored by the U.S. Army Engineer Topographic Laboratories.

Our main research topic has been the detection of moving objects and the estimation of the three-dimensional motion of the object, along with the estimation of the three-dimensional structure of the object. Within the context of the Autonomous Land Vehicle (ALV), motion estimation aids in detecting and tracking moving objects, tracking obstacles, and in determining the true motion of the ALV itself. A brief statement of the problem of motion analysis is: given a number of views of a scene, determine some correspondence between the views, and then determine the three-dimensional motion of objects in the scene. Methods differ on the meaning of correspondences, how many are required and in the formulation of the motion estimation equations.

Two basic approaches to motion analysis are the optical flow (short range) and feature point (long range) methods. Observations by psychologists, that the relative "flow" of scene points as projected on the retina can be used to determine the relative depth of objects, led computer vision researchers to the idea of optical flow. Rather than rely on direct computation of correspondences, these techniques use constraints on the smoothness of the surface (and thus the flow) and solve various equations that relate image values in consecutive images. These computations are limited to small motions between views and have proven to be unstable and unreliable in the general(real image) case. Optical flow methods are appealing in the ALV task for computing global (vehicle) motion since using all the flow data should average the errors out, but these methods have not been able to overcome their basic computational problems in this case. Sometimes these techniques are called short range methods since they assume that the position changes between views are small and that the views are closely spaced in time. This leads to a substantial computational load both in processing all the image points to solve for optic flow and in processing the large number of image frames.

The other methods, called feature point or long range techniques, attempt to solve many of the same problems using far fewer points in each image. These use a small set of corresponding points from the image sequence to compute the three-dimensional motion and structure. Different methods require different numbers of points in various numbers of frames under different assumptions. Generally, a set of equations, which encapsulate the constraints imposed by the assumptions (rigidity, small motions, etc.) are solved to derive the three-dimensional motion parameters. Often the formulations are very sensitive to noise in the input data and that makes the results unstable. Early approaches concentrated on using only two views of a scene, but these pairwise motion estimates must then be combined to get the true motion estimate. In order to capture the important constraints imposed by an extended sequence of views, we developed a technique to estimate the motion parameters using five frames for general motion and three frames for translational motion. This formulation has been given in the past annual reports and in [1].

We have adopted the feature based approach with more than two frames for most of our motion work, but have also explored some other techniques. Our effort has been in all aspects of feature-based motion analysis, including feature extraction, feature matching, motion estimation, and system integration. Our major effort the past year has been in feature matching, especially edge-based contours, with continuing work in developing a more integrated complete motion system. Other work includes spatio-temporal analysis of closely spaced image sequences and a further development of the multiframe motion equations that may lead to some simplifications and an increase in generality to handle some accelerations in the motion. We have also increased the number of test sequences available for testing of the integrated system and all the subsystems.

The following sections discuss the developments for this past year in the four research areas in more detail. The first section describes our continuing effort in matching groups of adjacent edge points (contours). The past work used straight line approximations to the contour (segments) and worked on pairs of images. This technique has been extended to find matches of individual edge points on the contour, using the segment based contour matches for context, and to track these matches through many frames. The segment matching restricts the search area for the "chain matching" algorithm that is applied to the individual edge points. This multi-level approach combines the robustness of a segment matching technique with the precision of the individual edge point matching. This technique extends to allow tracking the matching points through a long sequence of frames and thus can provide substantial amounts of data to the motion estimation program.

The next section discusses our efforts in analyzing and extending the multi- frame motion estimation approach. The homogeneous coordinate representation for rigid transformations is extended by adding time as an explicit component. The coefficients of this matrix representation are the constant coefficients of a nonhomogeneous system of linear difference equations. The motion parameters are easily calculated from the matrix coefficients, which are computed from the system of difference equations using the corresponding points in a sequence of views of the moving scene. This representation captures not only constant motion (constant rotation and translation), but also some cases of accelerated or non-rigid motion, e.g. a constant deformation while translating, acceleration in the direction of the axis of rotation, or acceleration (in any direction) of an object that has only a translational component of motion.

The third section outlines our new results in the spatio-temporal analysis of image sequences. This work uses very closely spaced images to first derive the image-plane velocity of connected edge points (curves). The curves can be broken at occlusion boundaries based on changes in the velocity between two portions of the curve. Compared to other techniques, this method explicitly handles occlusions.

The fourth section discusses our methodology for building a motion system by integrating the set of existing programs. We also present some early results in this process. This work is a continuation of the basis integration effort described in the past report with more emphasis on the generality of the design, by allowing for several different feature extraction and matching techniques.

The final section of this report describes our plans for future research. These plans involve the further development of the contour matching system, more efforts at integration and continued testing of all subsystems on more image sequences.

This report describes the work of several researchers in our group. The contour matching work was done by Salit Gazit with Gerard Medioni. The analysis and extension of motion estimation was performed by Wolfgang Franzen. The spatio-temporal work is by Shou-Ling Peng with Gerard Medioni The system integration work is by Keith Price and Igor Pavlin.

References

[1] Hormoz Shariat. *The Motion Problem: How To Use More Than Two Frames.* PhD thesis, University of Southern California, October 1986. USC IRIS report 202.

2 CONTOUR CORRESPONDENCES IN DYNAMIC IMAGERY

Motion analysis is an important research area within the field of Computer Vision, and plays a central role in biological systems. Sophisticated mechanisms for observing, extracting and utilizing motion exist even in simple animals. Processing image sequences via computers has various applications in the medical, biological, industrial, military and other fields. Several approaches have been tried for computational analysis of motion from image sequences, and many of them need a set of matching points or matching features for the motion analysis. Therefore matching features between consecutive frames is an important step in motion analysis. This problem is more difficult than model or stereo matching which assume additional constraints such as shape preservance or epipolar lines, since objects may move, change shape, disappear, *etc.*

This paper is devoted to the problem of identifying corresponding points in two time varying images of a moving object (or objects). We assume that the maximal distance between corresponding features is known, to restrict the search space.

The major difficulty in matching arises due to the need for making *global correspondences.* A local point or area in one image may match equally well with a number of points or areas in the other image. These ambiguities in local matches can only be resolved by considering sets of local matches globally and imposing some preference criterion.

The various matching algorithms differ in the primitives used for matching, the method used for local matching and the method used for global matching, if any. The basic primitive in our algorithm is a *section* of a *super-segment*, where a super-segment is an object dually defined as both a connected list of edgel points and a connected list of line segments, and a section is some arbitrary portion of a super-segment. We use segment matching merely as an initial guide to section matching, so unlike other segment matching algorithms [1,2,3], we are able to use important features such as *continuity* along the super-segments and sections of arbitrary (not only linear) shape and length for matching.

For local matching we use shape similarity between sections of super-segments and for global matching we use relaxation in the translation space. We believe that using sections of super-segments removes many of the problems resulting from segment or edgel matching, since the *continuity* information can be better preserved than in segments, which have the collinearity constraint, or edgels which do not contain any continuity information and the area between sections is much more reliable than merely "similar orientation". Using sections of arbitrary shape yields, we believe, a better match than using only linear segments, since curvature implies a much stricter constraint on the match. We allow these sections to grow as long as the area between them remains small, so we get very long reliable matches, which correspond to object boundaries.

Section 2.1 describes previous methods, section 2.2 contains the description of the algorithm, and section 2.3 presents the results and conclusions.

2.1 Existing Methods

2.1.1 Area based methods

Given two gray-level images, one would like to find a corresponding pixel for each pixel in each of the images, but the semantic information conveyed by a single pixel is too low to resolve ambiguous matches, so it becomes necessary to consider an area or neighborhood around each pixel. Three types of schemes can be found:

Differencing Schemes ([4,5,6,7] and others), a simple and fast method which is widely used. These systems tend to fail if the motion is small, illumination is not constant or the moving object is not easily distinguishable, and can be confused by noise.

Correlation Schemes were applied to measure cloud motion [8], traffic control [9] and to radar images [6]. They tend to fail in featureless or repetitive texture environment, are confused by the presence of surface discontinuity in the correlation window, are sensitive to absolute intensity, contrast and illumination and their complexity heavily depends on the size of the correlation window.

Gradient Schemes [10,11,12] are widely used for calculation of optical flow, and assume that the motion between successive images is very small, so they are very sensitive to noise.

2.1.2 Feature based methods

These systems match features derived from the two images rather than the intensity arrays directly. The commonly used features have been edgels, linear line segments and corners (points of high curvature). These systems are usually faster than area based systems since they consider much fewer points, yet preserve significant points. On the other hand a lot of pre-processing is needed to extract the features, and due to the sparse data these systems do not produce a dense matched map. Existing methods include graph matching techniques [13], relaxation [14,15], region matching [16] (useful when there is a significant change between frames, but tends to fail when there is occlusion) and more.

Matching edgels suffers from some of the limitations of the area based systems, since edgels are still very low-level. One isolated edgel is not very distinguished, so groups of edgels need to be taken in order to disambiguate matches.

A *line segment* (or just *segment*) is a linear approximation of connected edgel points and as such has some continuity information inherent to it, yet is local enough, so that the chance that a segment belongs to two physical objects is very small. Each segment contains information about its length, direction and position. Line segments are easy to represent and manipulate. Systems which match line segments exist mostly for Stereo image processing [2,3,17] and for image-model matching [1]. The main idea in these systems is essentially to locally slide two descriptions over each other for maximal fit. This approach guides our algorithm also.

2.2 Description of the Method

2.2.1 Primitives

We believe that the feature-based correspondense schemes have strong advantages over area-based schemes, because feature based systems consider much fewer points and are therefore faster than area based systems. By using features such as edgels, curves obtained by spatially linking these edgels, or even some approximation of these curves, the system is less susceptible to errors resulting from noise, change in illumination, *etc.* Curves formed of connected edgel points usually correspond to object boundaries, so the reduction in the amount of information does not necessarily mean reduction in the quality of the information.
Edgels however seem too local to be chosen as primitives. The advantages of *line segments* were already discussed. We use segment matching as an initial step in our algorithm. When we try to evaluate matches however, the disadvantages of segments come into view: they are at best only approximations of the "actual" curve and sometimes a bad approximation (a circle for example). A curve may be broken into segments differently depending on the segment fitting algorithm and on the amount of noise. The exact position of a match is not known because the segment matcher can only tells us that a line segment matches some other line segment but not which *pixels* actually match. Also most segment matching algorithms do not use the continuity between the segments. For these reasons, we have decided to use curves or *super-segments* which are objects each having an ordered list of segments belonging to the super-segment and a description of its curve as a chain of the "actual" points of the super-segment. Also each segment knows which super-segment it belongs to and the position in the super-segment chain where it fits. An example of this "dual representation" is given in figure 2-1.

The input is obtained by computing zero crossings [18] of convolution with Laplacian of Gaussian masks [19] to get the edgels, then link the edgels and finally fit curves by piecewise linear segments [20]. The curves produced in this method are long, closed and relatively not noise sensitive, but their locations and shape may not be accurate (as

Sa - A super-segment
{a1, a2, a3, a4} - its segments.

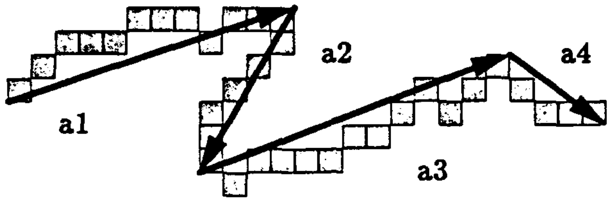


Figure 2-1: Example of a super-segment

explained in [19]).
Since continuity plays a very important role, we prefer zero crossings to other alternatives such as edgels produced by step masks [21] which have more accurate location, but the curves they produce are usually shorter and more noise sensitive. Another reason for using zero crossings of LoG masks is the fact that in moving from a large mask to a smaller mask we get additional edgels but no edgels disappear, which may be useful for a top-down approach.

2.2.2 The matching algorithm

The following is the general outline of our algorithm; the details are presented in the next subsections.

We match segments initially to obtain initial section segment matches, then divide each section point list into "pieces" (or sub-sections) and search for a best fit piece for each, trying to extend these pieces in the process. We evaluate these matches using relaxation in the translation space, and then remove overlapping (non-unique) matches, based on similarity in both shape and translation.

The Matching Algorithm

1. For each line segment in one image, find a subset of segments in the other image that can match this segment. (See section 2.2.3)

2. Match super-segments sections based on similarity in shape:

   (a) For every pair of maximal connected matching segment lists, define an initial match as the initial sections corresponding to these segment lists.
   (See section 2.2.4, step 1)

(b) Divide each (left) section into smaller pieces, and find for each piece the "most similar" piece in any matching section.
(See section 2.2.4, step 2)

(c) Extend each match by adding adjacent points to the pieces matched, so that the similarity error measure is minimized.
(See section 2.2.4, step 3)

3. Relaxation step: Remove matches for which not enough support exist, iterating until no matches are removed. (See section 2.2.5)

4. Remove overlapping matches. (See section 2.2.6)

5. Repeat once again steps 3, 2c and 4 (in that order).

In the next sections we discuss some of the steps in more detail.

Notation

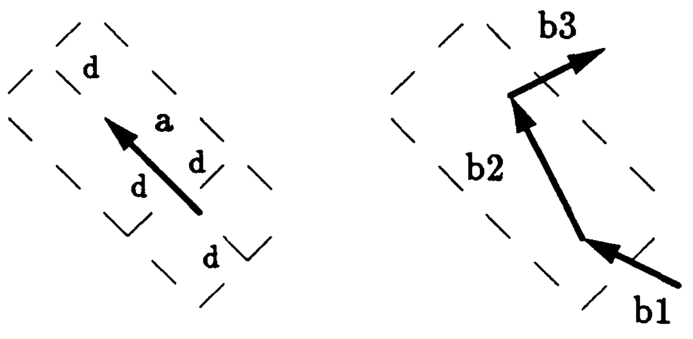
Let IMAGE_1 and IMAGE_2 be the images to be matched,
$A = \{a_i\}$ be the set of segments in IMAGE_1,
$B = \{b_j\}$ be the set of segments in IMAGE_2,
$S_A = \{s_{a_i}\}$ be the set of super-segments in IMAGE_1
and $S_B = \{s_{b_i}\}$ be the set of super-segments in IMAGE_2.
We use this notation since S_A (S_B) is actually a partition of A (B).
Also let the *maximal disparity* d be the maximal distance two corresponding features may have (measured in pixels).

2.2.3 Matching segments

The following algorithm computes for each segment $a_i \in A$ a subset of segments $b_j \in B$ that can match a_i:

For each segment $a_i \in A$ define a window $w(a_i)$ in which corresponding segments from B must lie, and define a similar window for segments in B. We have used a rectangular window parallel to the segment with width $2d$ and height $2d + l_i$, as $w(a_i)$. Note that $b_j \in w(a_i) \Rightarrow a_i \in w(b_j)$.

Let $a_i \in A, b_j \in B$ be two segments with orientations θ_i, θ_j and length l_i, l_j respectively. We say that a_i matches b_j if the following conditions hold:



Left segment a matches right segments {b1,b2},
but not b3 (orientation). d is the maximal disparity..

Figure 2-2: Example of matching segments

$b_j \in w(a_i)$, a_i and b_j have "similar" orientation (the similarity measure is defined by equation 1), and the middle point of the shorter segment must intersect the window of the other segment.

$$|\theta_i - \theta_j| \leq \theta + \frac{\pi}{2} \cdot l \cdot \left(\frac{1}{l_i} + \frac{1}{l_j}\right) \tag{1}$$

θ and l are constants. We used $\theta = \frac{\pi}{6}$ and $l = 1$. (See [22]). Figure 2-2 contains an example of matching segments.

2.2.4 Matching super-segments based on shape similarity

Note: Depending on the context, a *super-segment* is an ordered list of segments or an ordered list of edgels comprising the segments.

Definition 1 *The* position *of a point in a super-segment is the arc length of the point.*

Definition 2 *A* section *of a super-segment is a connected list of edgels, which is a part of the super-segment (Note that segments and super-segments are a special case of section). A* piece *is a portion of a section.*

See figures 2-3 and 2-5 for an example.

The following algorithm computes initial section matching based on similarity in the shapes of the sections:

1. Initially two super-segments s_{a_i} and s_{b_j} can match if any of their segments match, and for each super-segment s let $S_p(s)$ be the set of its possible matching sections,

which are simply the maximal consecutive sub-chains which segments correspond. (see figure 2-3).

2. For every pair of matching sections (P, Q), divide P into pieces, so that $P = \{p_1, p_2 \ldots, p_k\}$. Using the segment matches, find a corresponding piece q_i to every piece p_i (in the same fashion as before). Note that the "actual match" for p_i is probably contained in q_i, and therefore we need to search for it. We "slide" p_i along q_i searching for the match with the lowest similarity error measure.

   The similarity error measure is the *area of the match* over the total number of points in the two matching pieces squared. Figure 2-4 illustrates the idea. Appendix A contains a description of an efficient algorithm to compute the area between two matching sections.

3. For each match (p_i, m_j) try to "extend" it by adding neighboring points as long as the error (computed in the same way as above) decreases. To reduce time complexity we used a binary search type extension (see figure 2-5).

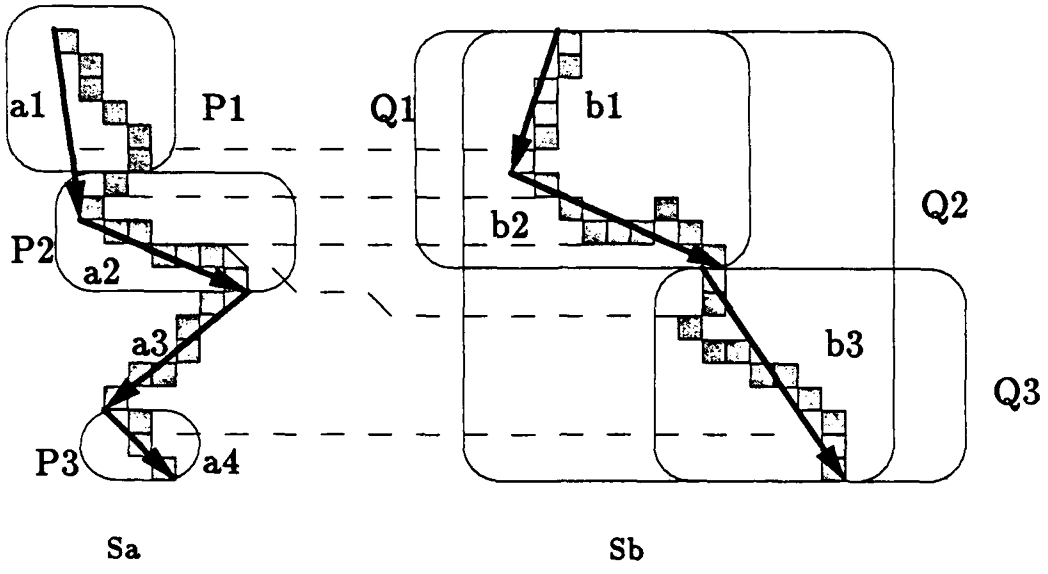
Notes

Matching each piece is done independently, so non unique matches are allowed, since we hope that at least one will "catch" its correct location. Dividing the initial large section into smaller pieces is necessary since the sections often do not fully match, but portions of them do (due to motion of objects, changes in illumination, occlusion or errors of the edge detector). Extending the matches is necessary, as long matches are much more reliable than shorter ones, so good matches are better distinguishable from bad ones.

We chose a bottom up approach, in which we break the initial matching sections into small enough pieces and try to match each such section, then try to extend the match as long as the shape of the curve is similar enough (Another option is to determine where is the best place to "break" a super-segment, but this is complicated, since it requires finding corners, junctions and other high level features, and may fail when we have occlusion and motion). The size of the initial pieces was chosen as $\frac{l}{\log(l)}$, where l is the length of the shorter of the two initially matching sections. This was a compromise figure between having a constant number of pieces per section (which penalized long sections) and having a constant piece size (which penalized short sections).

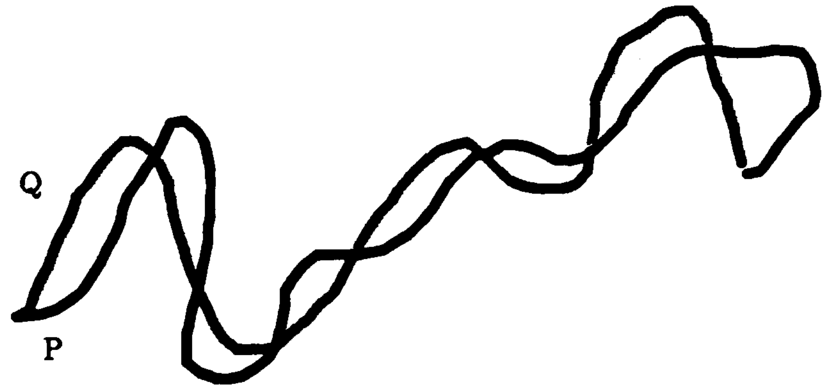
2.2.5 Relaxation step

In this step we evaluate matches using a global criterion, namely similarity in 2-D translation of neighboring matches (similar to other matching algorithms). We discard a



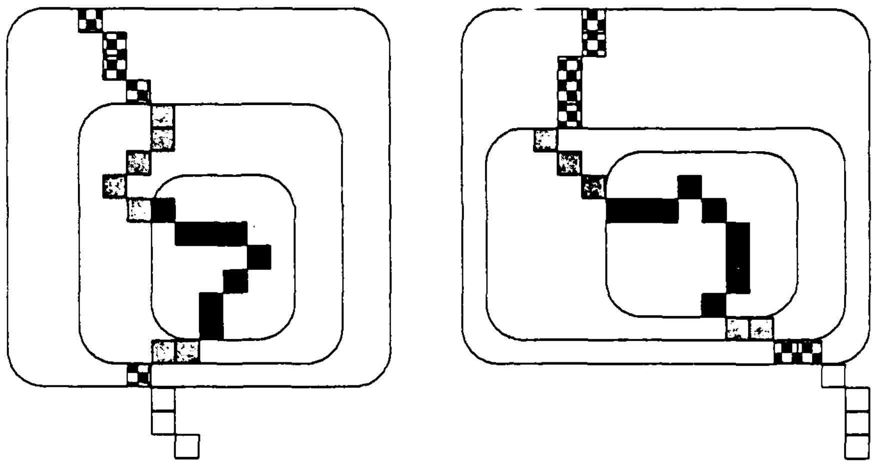
Sa, Sb - super-segments
{a1,a2,a3,a4} - segments of Sa, {b1,b2,b3} - segments of Sb.
P1, P2, P3 - sections of Sa, Q1, Q2, Q3 - their initial matching sections.
Matching is based on segment matching : a1-b1, a1-b2, a2,-b2, a2-b3, a4-b3.

Figure 2-3: Example of matching super-segment sections



Score of the match (P, Q) is the area between them (colored) over the total number of points in the two sections squared.

Figure 2-4: Area between two sections (P, Q) (Q translated to start where P starts)



Extension of the match:
Initially the inner sections only match, then they are extended until the score becomes worse.

Figure 2-5: Example of matching super-segment sections

match if the total number of points in matches which support it is below some threshold value (defined later). The support is based on "similar" average translation within some neighborhood.

Definition 3 *s_{a_k} is a* neighbor *of s_{a_i} if the distance between their closest points is less than the maximal disparity.*

The neighbors can be computed in the same way as the initial matches were computed in the previous section.

Let d_e correspond to the expected error in the "real" position (after compensating for the motion) of the object (d_e should be 0 if no rotation, expansion or errors of the edge detector occur, but is usually larger). We used σ (the space constant of the LoG filter) when we did not expect a major change in the shape (due to expansion), since the error in the position of the edgels depends on σ. Otherwise we used the maximal disparity (supplied by the user).

Let $M_{ij} = (p_i, m_j)$ be some match with with translation $(\overline{x_{ij}}, \overline{y_{ij}})$ where p_i is a section of a super-segment s_{a_i} and m_j is a section of a super-segment s_{b_j}. A match $M_{hk} = (p_h, m_k)$ can *support* M_{ij} if $|\overline{x_{ij}} - \overline{x_{hk}}| \leq d_e$ and $|\overline{y_{ij}} - \overline{y_{hk}}| \leq d_e$, it is not too short (its length is at least σ), either s_{a_h} is a neighbor of s_{a_i} or s_{a_i} has no neighbors and either s_{b_k} is a neighbor of s_{b_j} or s_{b_j} has no neighbors. Note that M_{ij} can support itself. M_{ij} is kept if the total length of the matches that can support it is above a certain threshold or one of these matches is long enough. (our threshold was half the sum of the average length of the matches and the length of the longest match, and a match was long enough to support alone if its length was at least 2σ.)

We iterate until no matches are removed.

2.2.6 Removal of overlapping matches

Let $M_1 = (P, Q)$ and $M_2 = (O, R)$ be two matches. We say that M_1 and M_2 overlap if either P and O are sections of the same left super-segment s_{a_i}, and have points in common, or Q and R are sections of the same right super-segment s_{b_j}, and have points in common.

Assume (w.l.o.g.) the first case, then two possiblities exist:

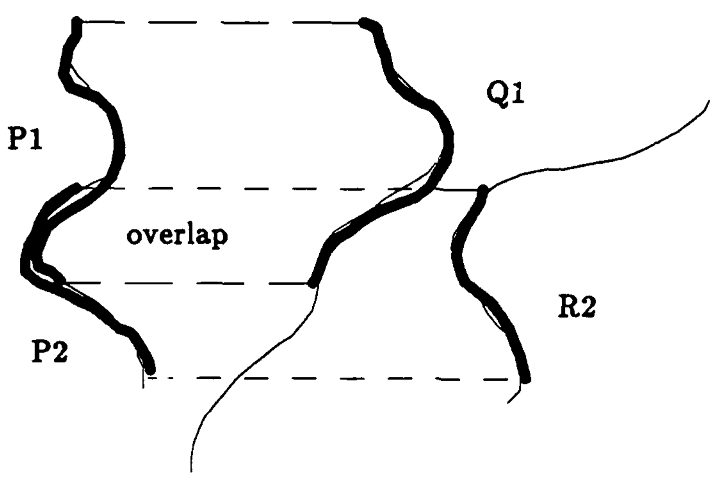
- Partial overlap
- Complete overlap.

The two cases are illustrated in figure 2-6. In both cases the solution is to take the better match and the remainder of the other match. In the example of figure 2-6, we take matches M_2 and the remainder (the non overlapping portion) of M_1. In the second case, we prefer M_3.

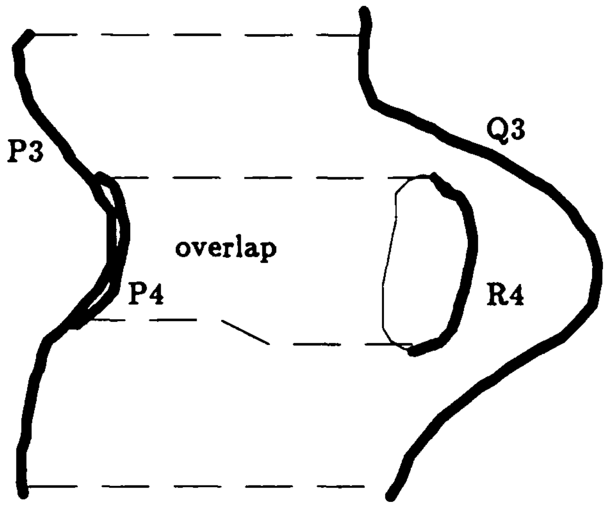
To evaluate matches we try to use both the similarity in shape and in translation. We say that a match M is better than a match N if the score of M (as computed by the previous step) + (1000 over the number of points in supporting matches) is lower than that of N.

2.2.7 Why repeat the previous steps ?

In the algorithm to match sections based on shape similarity, each section was matched and extended independently. Therefore we expect a lot of overlapping matches. For example, consider the case of matching two identical super-segments of length l: we divide the first to $\log(l)$ sections and then extend each independently, so we end up with $\log(l)$ identical matches. The overlap-removal algorithm will remove $\log(l) - 1$ of these matches. In all our experiments the number of matches was significantly reduced after this step. If many matches were removed, there can now be matches with not enough support (see section 2.2.5), so we need to apply the relaxation again. If two overlapping matches were divided by the overlap-removal algorithm, and then one of them was removed by



M1 = (P1,Q1) partially overlaps M2 = (P2,R2)



The match M3 = (P3,Q3) contains the match M4 = (P4,R4).

Figure 2-6: The two overlap possibilities

the relaxation, the remaining one can now be extended again. Therefore we apply the step to extend matches again and then remove the new overlaps created by extending the matches.

Theoretically we can then repeat the relaxation again, then extending and so on. However, the changes at this stage are expected to be minor, and since we cannot guarantee convergence, we do it only once.

2.2.8 Combining matches along the sequence

Matching two images is useless if a way to combine matches along the sequence is not offered. Most matching algorithms ignore the problem, though for some (edgel matching algorithms) the solution is straight forward.

In our case, as already mentioned, we have section matches for sections of arbitrary length and shape. If section P_1 (in frame 1) matches section P_2 (in frame 2) and section Q_2 (in frame 2) matches section Q_3 (in frame 3),then sections P_2 and Q_2 must have one of the following relations: Either they have no points in common, one of them is a sub-section of the other, or they partly overlap. Combining these matches is, of course, possible only in the last two cases. This problem is very similar to the overlapping of matches discussed previously. Indeed these two cases are partly illustrated in figure 2-6. Our solution is fairly similar too. We compute the overlapping sub-section of P_2 and Q_2, say R_2 and then find R_1, the sub-section of P_1 (in frame 1) that best matches R_2. R_3 is computed in a similar way. The result is a match (R_1, R_2, R_3). This process can be iteratively applied to obtain multiple matches $(M_1, M_2, \ldots, M_k)$ for k frames. To eliminate bad multiple matches (if a pair of matches was erronous, the whole multiple match is incorrect), the variance in the 2-D translations between successive frames is threshoded.

We applied this simple algorithm to the sequences in the results section and it seems to perform well.

Problems with this method are mainly that it can only handle sections that match throughout the sequence. Disappearance of points (due to occlusion or disappearance of objects from the images) cannot be handled by this simple algorithm. In addition, this method will discard a long multiple match if one of the matching pairs is wrong (and so the variance between matches is large). It would be better to detect this error instead. We are working now on extending the method to handle these problems.

2.3 Results

We applied our algorithm on a number of real images, indoor as well as outdoor scenes. As long as the shapes of objects in the scene (as projected in the image) did not change much, the results were very good. The results are shown by displaying only those points for which a match was found, and drawing an arrow to the closest point in the other section (after translating to start at same location). The arrow is drawn for every fifth point in a matching section for each matching section), for clarity.

We give five examples, in each of which
subfigures (a),(b) contain two matched original images,
subfigures (c),(d) contain the super-segments obtained from the zero crossings of the convolved images,
subfigure (e) contains the result of the matching
and subfigure (f) contains result of combining matches along the sequence.

1. Figures 2-7 contains two 512 × 512 pixels images taken from a sequence of a road scene ($\sigma = 10, d = 25$). Both the observer and the other car are moving. Subfigure (f) shows result of combining the sequence along 91 frames, actually matching only every sixth frame (so combining 15 matches).

2. Figures 2-8 contain two 512 × 512 pixels images of a car crossing the observer view-point ($\sigma = 10, d = 30$). The algorithm performed well on the image, even though the disparity was large, which shows that the location of the match does not matter much, as long as the *shape does not change* significantly between the frames. Subfigure (f) shows result of combining matches along 5 frames.

3. Figures 2-9 contains two 256 × 256 images of an office scene ($\sigma = 5, d = 10$). The camera faces the direction of motion, so we expect objects to expand. Subfigure (f) shows result of combining matches along the sequence, using 26 frames but matching only every fifth frame (5 matches).

4. Figures 2-10 contains two 256 × 256 images of a corridor ($\sigma = 5, d = 10$). The camera faces the direction of motion, so we expect objects to expand. Subfigure (f) shows result of combining matches along the sequence, using 16 frames but matching only every fifth frame (3 matches).

5. Figures 2-11 contain two 256 × 256 images of an outdoor scene of trees ($\sigma = 5, d = 10$). This is a lateral motion case, which is made hard by the large disparity differences, and therefore most stereo algorithms will not match it successfully. We did not use the knowledge that motion is only lateral and allowed search in all directions, yet the algorithm was able to match the scene quite well. Using the epipolar constraint would probably improve the result.

The images in Figures 2-9, 2-10 and 2-11 were obtained from SRI International, courtesy of Dr. Bolles.

2.4 Conclusions and Future Work

We have shown an algorithm to compute correspondence between 2 frames with very few constraints. We suggested the use of super-segments and sections of super-segments. Correspondence was based on shape similarity between matching sections and on translation similarity between matches, and demonstrated some results on a number of real images.

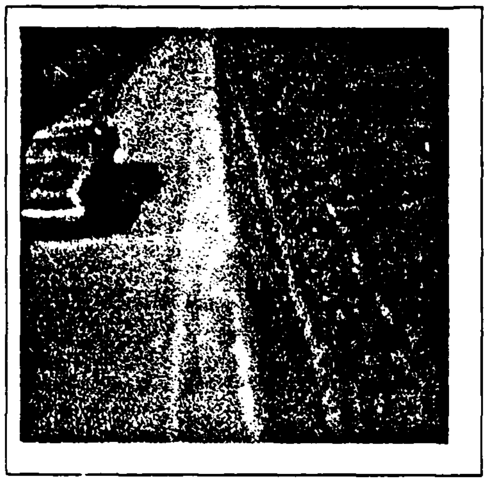
The advantages of our method were discussed in the previous sections: the use of continuity and sections of arbitrary shape and size in matching, the use of length of a match, evaluation of matches based both on shape and on common translation.

Some comments are in order:

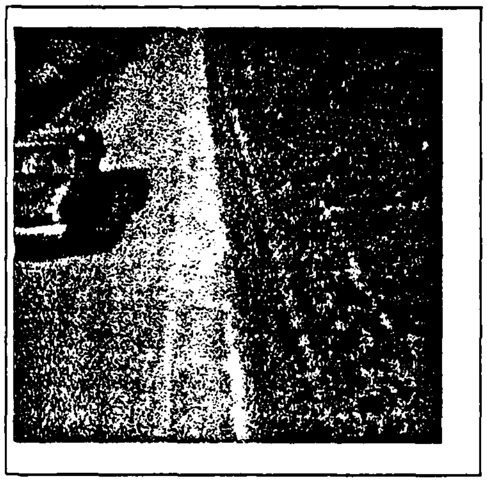
- The algorithm has a very heuristic flavor.
- The algorithm performs best on long curved contours, so it seems to best fit for matching zero crossings curves or region contours. We plan to try applying it to regions and to curves of the same image, processed with different LoG masks.
- We may get better results for stereo pairs by applying this algorithm with the epipolar constraint, as it can handle sharp changes in disparity, as demonstrated by example 2-11.
- The figure computation is made in 2-D only, but we can find the actually corresponding points using areas of high curvature or even the simple method we used for displaying the results (a left point matches the closest point in the translated matching right section). These point-to-point matches can be used for motion estimation in 3-D. We are currently working on using the Motion Estimation algorithm developed in [23]. This algorithm uses matching points in three or more frames to estimate 3-D motion and location of points in frames as well as give some error measure to the match. Since using the Motion Estimation algorithm requires matches in multiple frames, an algorithm to combine the results of matching pairs of images will be useful.

A Computing area between two sections

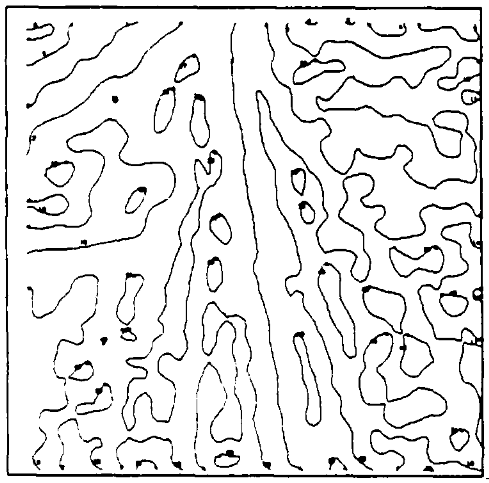
The following is a general idea of the computation of area between two matching sections (some of the details have been left out). The idea is to translate the right section to have same starting point as the left section, and add points to ensure that the last point is



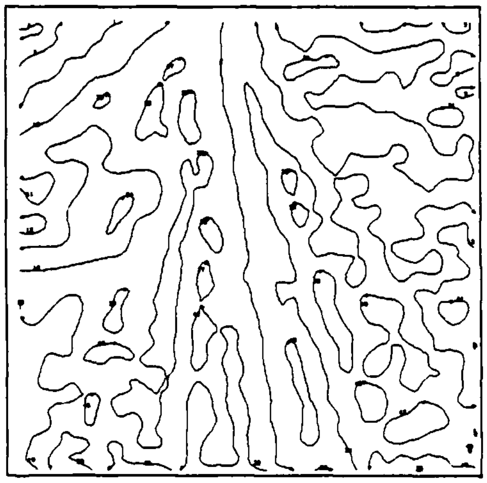
(a) Frame 1



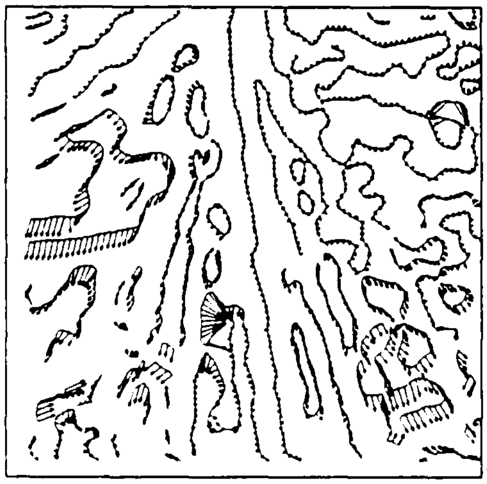
(b) Frame 2



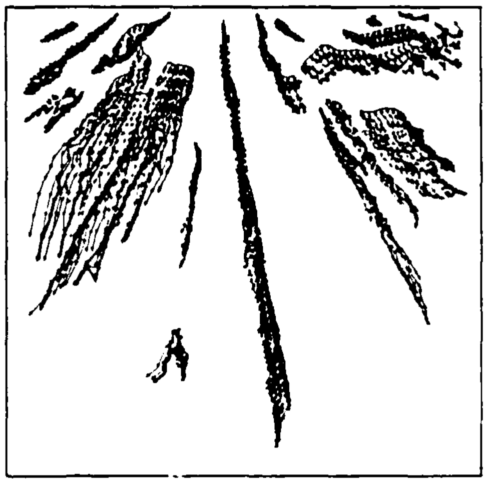
(c) Zero crossings of (a)



(d) Zero crossings of (b)



(e) Matches of (c),(d)

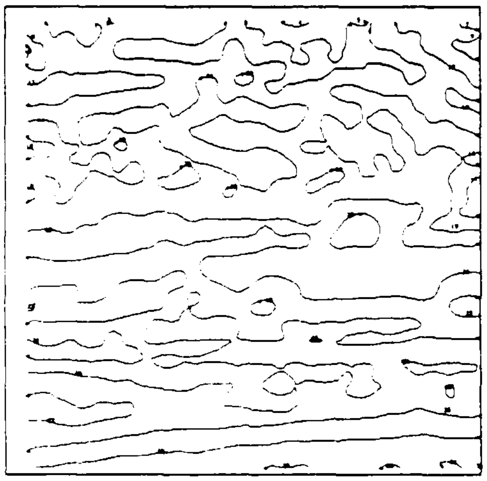


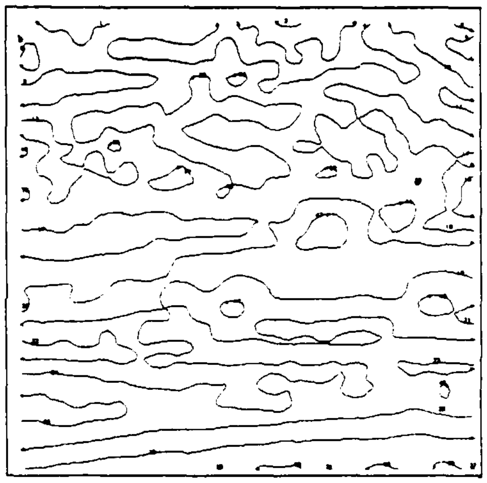
(f) Multiple matches

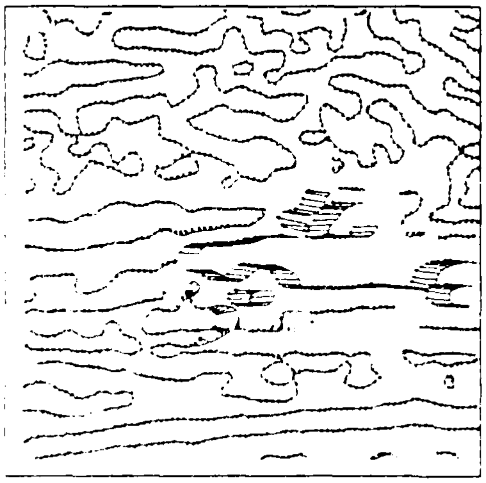
Figure 2-7: Advancing car

(a) Frame 1

(b) Frame 2


(c) Zero crossings of (a)


(d) Zero crossings of (b)


(e) Matches of (c),(d)

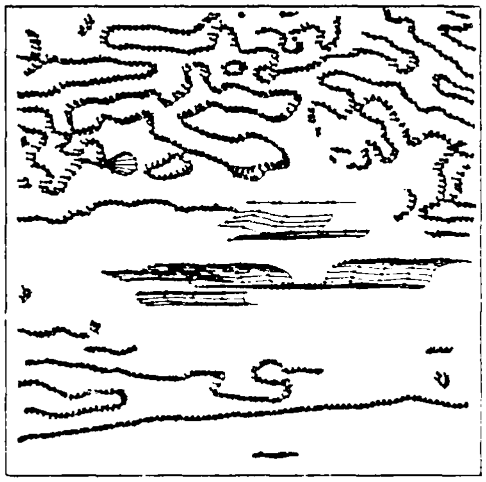
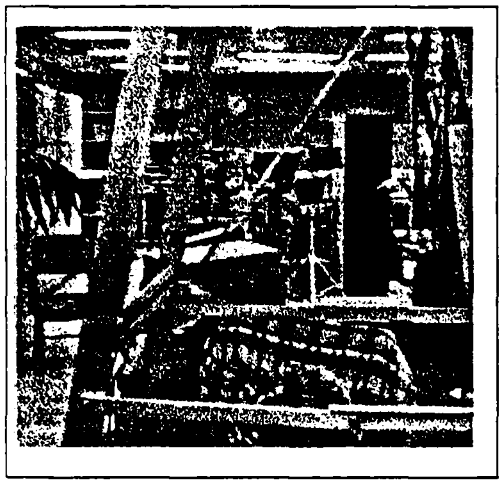
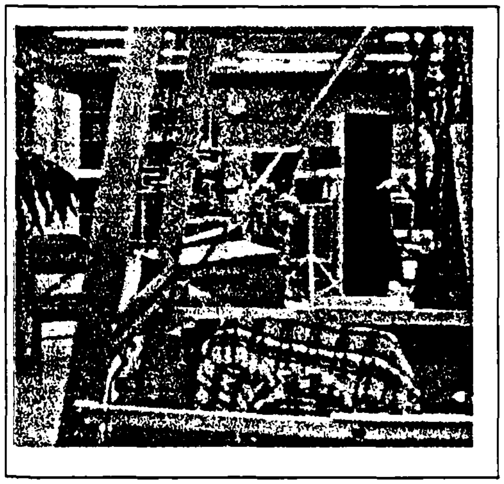

(f) Multiple matches

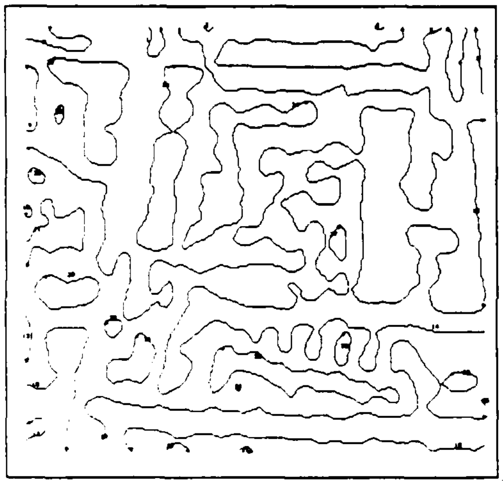
Figure 2-8: Crossing car



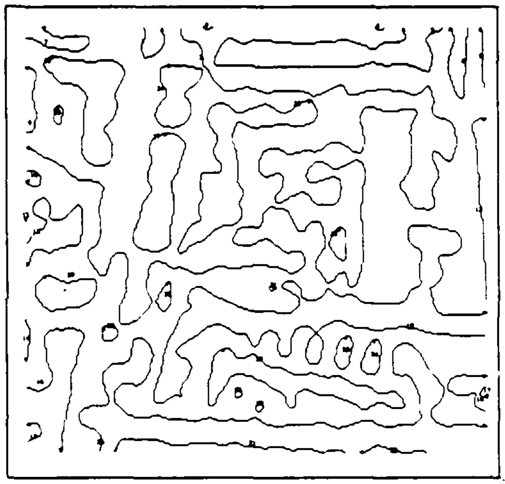
(a) Frame 1



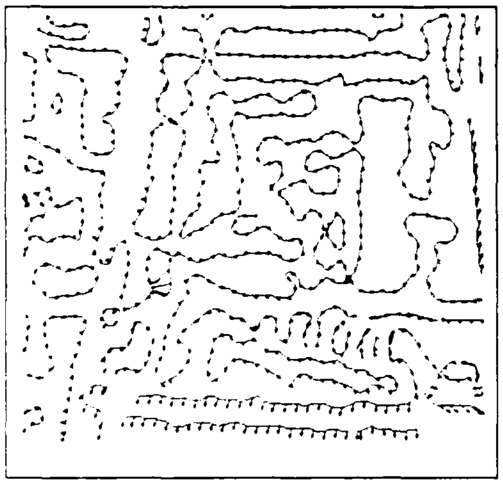
(b) Frame 2



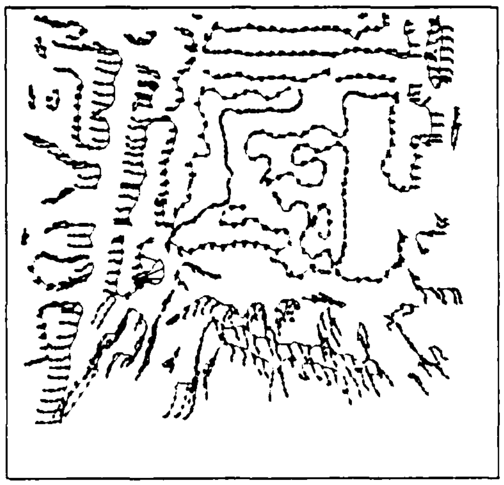
(c) Zero crossings of (a)



(d) Zero crossings of (b)

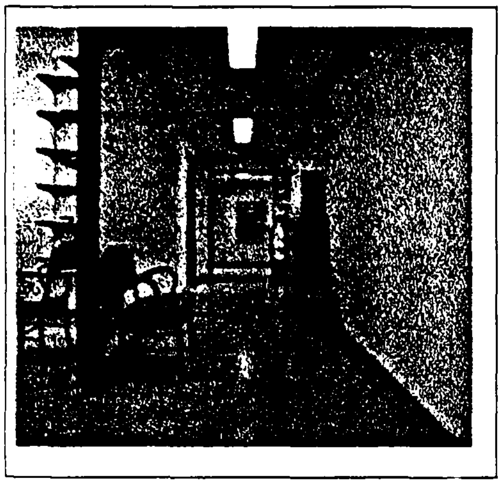


(e) Matches of (c),(d)



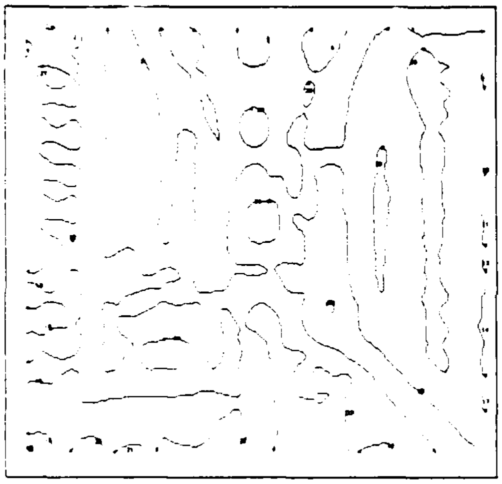
(f) Multiple matches

Figure 2-9: Office Scene

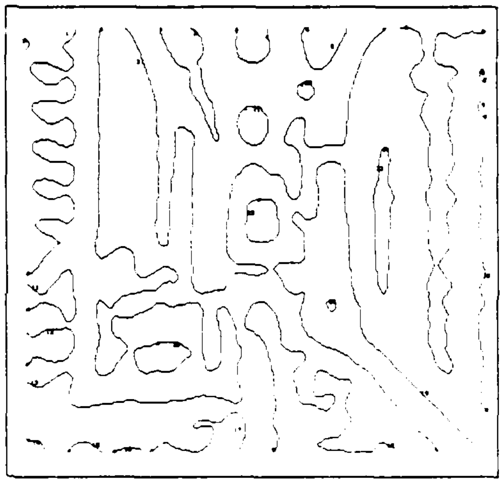


(a) Frame 1

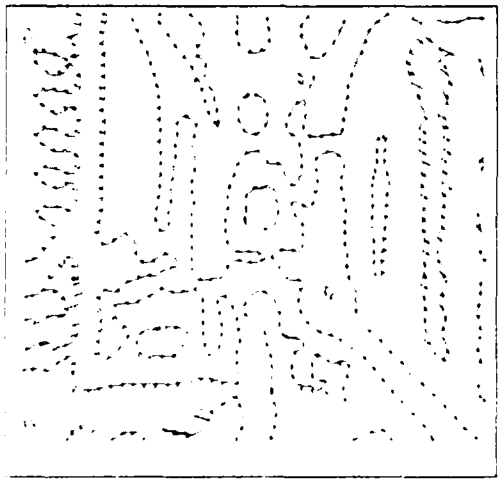
(b) Frame 2



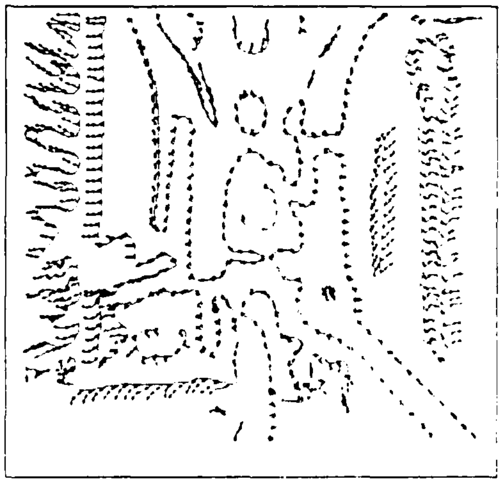
(c) Zero crossings of (a)



(d) Zero crossings of (b)



(e) Matches of (c),(d)

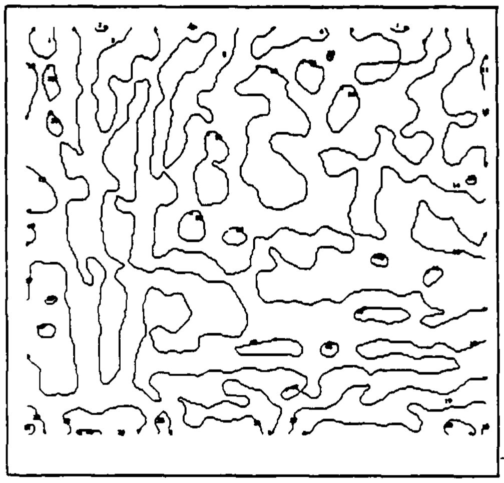


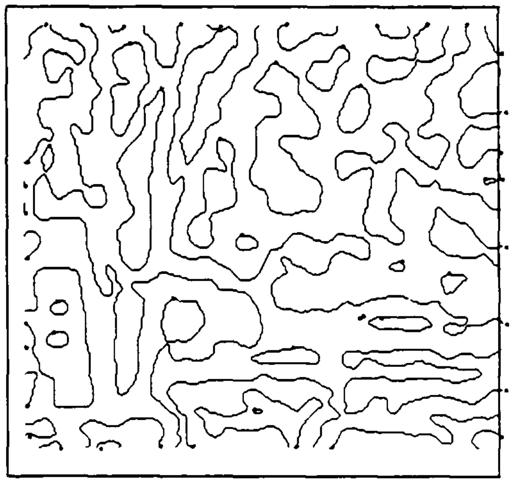
(f) Multiple matches

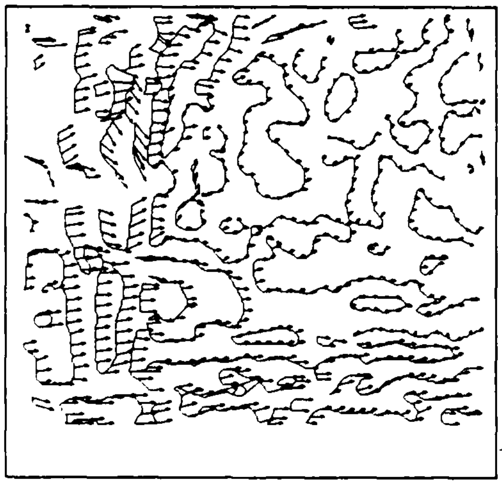
Figure 2-10: Hallway

(a) Frame 1

(b) Frame 2


(c) Zero crossings of (a)


(d) Zero crossings of (b)


(e) Matches of (c),(d)

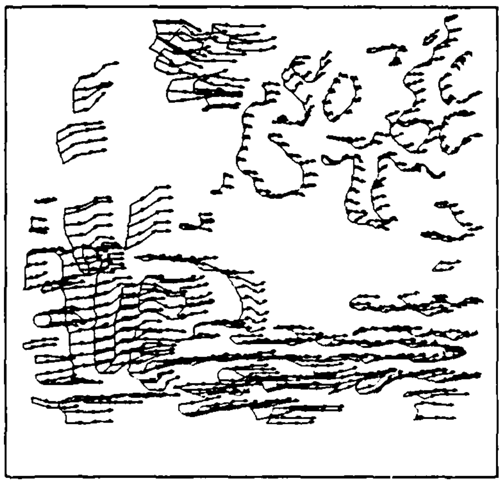

(f) Multiple matches

Figure 2-11: Outdoor Scene (trees)

also the same (we might have to remove remaining points from the sections). We get two sections which start and end points are the same, so we have a *cycle*. We find all simple cycles (cycles which do not cross themselves) and compute the area for each by a simple procedure. Figure 2-4 illustrates the idea.

The algorithm:

Assume sections $P = (P_1, P_2, \ldots, P_k)$ and $Q = (Q_1, Q_2, \ldots, Q_l)$ are possible matches, where $P_i = (x_i^p, y_i^p)$ and $Q_i = (x_i^q, y_i^q)$. Translate Q to start at P_1. Define an *intersection point* as a point P_i such that there is a point Q_j in the translated Q, such that $P_i = Q_j$ (Note that $P_1 = Q_1$). Find all intersection points (this can be done linearly by drawing the left section in the plane and traversing the translated right section). The points of the two sections which lie between two adjacent intersection points form simple cycles. The sum of the areas of the simple cycles is the area of the match. We compute it by assigning every cycle point (x, y) a score $s = x_n - x_p$ where x_n and x_p are the x coordinate of the next and previous points on the cycle. Let $A[x] = ((y_1, s_1), \ldots, (y_r, s_r))$ a sorted scan line. The area of the cycle along the scan line is the sum of all distances for which the accumulated score is non-zero.

The algorithm is linear in the number of points of the two sections, except where we sort the rows. This step requires $r \log(r)$ time, where r is the number of points of this scan line. It will usually be a constant though, since points along horizontal lines have score zero and therefore do not affect the sum and can be eliminated.

References

[1] G. Medioni and R. Nevatia. Matching images using linear features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):675–685, Nov 1984.

[2] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, 1985.

[3] N. Ayache and B. Faverjon. A fast stereovision matcher based on prediction and recursive verification of hypothesis. In *In Proceedings of the 3rd Workshop on Computer Vision: Representation and Control*, pages 27–37, Bellaire, Michigan, Oct 1985. IEEE.

[4] R. Jain, D. Militzer, and H.-H Nagel. Separating non-stationary from stationary scene components in a sequence of real world tv-images. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 612–618, Cambridge, Mass, Aug 1977.

[5] R. Jain, W. N. Martin, and J. K. Aggarwal. Segmentation through the detection of change due to motion. *Computer Graphics and Image Processing*, 11(1):13–34, Sep 1979.

[6] R. Jain and H.-H Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):204–214, Apr 1979.

[7] R. M. Onode, N. Hammano, and K. Ohda. Computer analysis of traffic flow observed by substractive television. *Computer Graphics and Image Processing*, pages 377–399, Sep 1973.

[8] J. A. Leese, C. S. Novak, and V. R. Taylor. The detection of cloud pattern motions from geosynchronous satellite image data. *Pattern Recognition*, 2:279–292, Dec 1970.

[9] K. Wolferts. Special problems in interactive image processing for traffic analysis. In *Proceedings of the 2nd International Joint Conference on Pattern Recognition*, volume 1,2, 1974.

[10] B.K.P. Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[11] C. Cafforio and F. Rocca. Methods for measuring small displacements of television images. *IEEE Transactions on Information Theory*, 22(5):573–579, Sep 1976.

[12] C. L. Fennema and W. B. Thompson. Velocity determination in scenes containing several moving objects. *Computer Graphics and Image Processing*, 9:301–315, Apr 1979.

[13] C. J. Jacobus, R. T. Chien, and J. M. Selander. Motion detection and analysis by matching graphs of intermediate level primitives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):495–510, Nov 1980.

[14] S. T. Barnard and B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, July 1980.

[15] L. Dreschler and H.-H. Nagel. Volumetric model and 3-d trajectory of a moving car derived from monocular tv-frame sequence of a street scene. In *International Joint Conference on Artificial Intelligence*, Vancouver, Canada, Aug 1981. IEEE.

[16] K. Price and R. Reddy. Matching segments of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):110–116, Jan 1979.

[17] R. Mohan, G. Medioni, and R. Nevatia. A fast stereovision matcher based on prediction and recursive verification of hypothesis. In *Proceedings of the 1st ICCV*. IEEE, 1987.

[18] A. Huertas and G. Medioni. Detection of intensity changes with subpixel accuracy using laplacian-gaussian masks. *Pattern Analysis and Machine Intelligence*, PAMI-8(5):651–664, Sep 1986.

[19] J. S. Chen and G. Medioni. Detection, localization and estimation of edges. In *Proceedings of Workshop on Computer Vision*, Miami Beach, Florida, Nov. 1987. IEEE.

[20] S. L. Gazit and G. Medioni. Accurate detection and linking of zero crossings. Unpublished, 1987.

[21] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, 1980.

[22] R. Nevatia and K. Price et al. Research in knowledge-based vision techniques for the alv program. Technical Report 201, IRIS, University of Southern California, Los Angeles, California, Sep 1986.

[23] H. Shariat. *The Motion Problem - How to use more than two frames*. PhD thesis, IRIS, University of Southern California, Los Angeles, California, Oct 1986.

3 NATURAL REPRESENTATION OF MOTION IN SPACE-TIME

The analysis of motion in time-varying imagery is an active research area in computer vision. There are a number of good surveys on the subject, such as [1].

Over the last few years, there has been an increasing trend of imposing constraints in addition to rigidity, such as constancy of motion, to facilitate the analysis of motion, or to solve the structure from motion problem. In his now classic work, Ullman [2] originally solved the structure from motion problem (orthograhic projection) for four points in three frames, using no assumption other than rigidity. The best recent work, that imposes additional contraints, is that of Shariat [3], who studied objects undergoing uniform translation and rotation. Among other cases, he solved the structure from motion problem (perspective projection) using only three points in three frames.

The approach presented in this paper is somewhat different. Rather than explicitly putting contraints on motion, we start with systems of equations whose solution leads to nontrivial, but mathematically tractable, classes of motion. The proper selection of such a system is a matter of physical and mathematical intuition. Rigid motion of the form described in this paper is more general than the motion studied by Shariat, and often allows the solution of the structure from motion problem for the same number of points in the same number of frames, as in his case. What is more important, this representation also allows the study of *structure from nonrigid motion.* With the notable exception of [4], [5], [6], and [7], relatively little work has been done on the quantitative analysis and representation of nonrigid motion.

In the following, we begin with a brief review of homogeneous coordinates. Then a generalization of homogeneous coordinates, that we call chronogeneous coordinates, is described. (The term chronogeneous is actually a contraction of *chrono*-homo*geneous*). After introducing some additional notation, we derive a vector equation that expresses the position of a point, at an arbitrary juncture in time, in terms of its initial position and the matrices describing the motion of the object and the motion of the camera. Then a characterization of chronogeneous motion is given, with particular emphasis on rigid motion. A novel result involving the recovery of absolute depth from a monocular image sequence is presented. Finally, we summarize what we believe to be the major contributions of this work, and discuss future research.

3.1 Homogeneous Coordinates

Rigid transformations of bodies are typically represented using homogeneous coordinates. Homogeneous coordinates were introduced by Roberts in [8]. [9] also provides a good overview of homogeneous transformations. The usefulness of this representation stems form the fact that rigid transformation and perspective are expressible in matrix form. The homogeneous coordinate representation of the 3D point $(x, y, z)^T$ is any 4D point of the form $(\omega x, \omega y, \omega z, \omega)^T$ where $\omega \neq 0$. The value of the last component, ω, is normally taken to be 1, until a perspective projection operator is applied.

In the following, let $\vec{x}_{3D}$ be the 3D position of a point, and let $\vec{x}_{4D}$ be its corresponding homogeneous representation. Also, let $\vec{x}'_{3D}$ and $\vec{x}'_{4D}$ represent corresponding transformed positions of these points. A general homogeneous transformation may be expressed as

$$\vec{x}'_{4D} = \mathcal{H}\vec{x}_{4D}$$

where

$$\mathcal{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

is the homogeneous transformation matrix.

A general rigid 3D tranformation may be expressed as

$$\vec{x}'_{3D} = \mathcal{R}\vec{x}_{3D} + \vec{T}$$

where

$$\mathcal{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

is a rotation matrix, and $\vec{T} = (t_1, t_2, t_3)^T$ is a translation vector.

The same transformation is expressed more succinctly in homogeneous coordinates as

$$\vec{x}'_{4D} = \mathcal{H}\vec{x}_{4D}$$

where

$$\mathcal{H} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Despite the usefulness of homogeneous coordinates for representing a rigid transformation between two frames, this representation does not lend itself to describing motion in multiframe imagery. This is due to the fact that even for relatively simple kinds of motion, the homogeneous transformation matrix changes from one frame to the next. Some examples of simple types of motion that yield changing $\mathcal{H}$-matrices are:

- The ballistic motion of a (nonrotating) ball accelerating due to the force of gravity.
- The motion of a (nonrotating) camera on a uniformly accelerating vehicle.
- The motion of a spinning top that is rolling across the floor (assume no precession).
- The motion of a wheel of a car moving at a constant velocity, when viewed from the side.

Ideally, one would like to describe motion in such a way that the motion parameters corresponding to commonly occuring types of motion are constant over time. This motivates the following extension to homogeneous coordinates, which allows the natural description of the above types of motion, as well as other types, even nonrigid motion.

3.2 Chronogeneous Coordinates

This section describes a generalization of homogeneous coordinates in the time domain, which we call *chronogeneous coordinates*. The homogeneous coordinate representation is extended by augmenting it to encode time explicitly. The chronogeneous coordinate representation of the 3D point $(x, y, z)^T$ at time t is any 5D point of the form $(\omega x, \omega y, \omega z, t, \omega)^T$ where $\omega \neq 0$. The value of the last component, ω, is normally taken to be 1, until a perspective projection operator is applied. Note that the factor ω does not multiply the time component. Whereas the spatial components of a point, at least conceptually, range over a continuous set of values, the time component is discrete and only takes on values which are multiples of ΔT, where $1/\Delta T$ is the frame rate of the imaging system. The frame rate is assumed to be a known constant.

Except for the perspective projection matrix discussed below, we will consider only chronogeneous transformation matrices of the following form, which we call *standard* chronogeneous matrices:

$$\mathcal{C} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & \gamma_1 & p_1 \\ s_{21} & s_{22} & s_{23} & \gamma_2 & p_2 \\ s_{31} & s_{32} & s_{33} & \gamma_3 & p_3 \\ 0 & 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{S}} & \vec{\Gamma} & \vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{1}$$

Since almost all the chronogeneous matrices we discuss are standard, we often drop this designation.

The value of the element δt, of $\mathcal{C}$, is restricted to being an integer multiple of ΔT. In fact, the value will always be a known multiple of ΔT. Therefore, this representation has 15 degrees of freedom and, in general, represents nonrigid motion. We sometimes refer to the submatrix, $\mathcal{S}$, of $\mathcal{C}$, as the *structural deformation submatrix*, or simply the *deformation submatrix*. If $\mathcal{S}$ is a rotation matrix, then $\mathcal{C}$ represents a rigid transformation and has only 9 degrees of freedom. The subvector $\vec{\Gamma} = (\gamma_1, \gamma_2, \gamma_3)^T$ has units of velocity, but roughly encodes information about acceleration, and the subvector $\vec{P} = (p_1, p_2, p_3)^T$ has units of displacement, but roughly encodes information about velocity.

The matrix, $\mathcal{C}$, is equivalent to the following 3D transformation:

$$\vec{x}_{3D}(t + \delta t) = \mathcal{S}\vec{x}_{3D}(t) + t\vec{\Gamma} + \vec{P} \tag{2}$$

In addition, it causes the time to be incremented by the amount δt. If $\mathcal{S}$ is a rotation matrix, and if we drop the dependence on time and view $t\vec{\Gamma} + \vec{P}$ as the translation vector, then the above vector equation reduces to the general rigid 3D transformation. Vector equation (2) is a nonhomogeneous system of first order linear difference equations with constant coefficents. Therefore, the theory of linear difference equations may be used to study solutions of this equation.

3.3 Common Notation and Assumptions

This section defines some common notation and assumptions that are used throughout the remainder of the document.

3.4 Special Chronogeneous Matrices and Operators

This section introduces some often used chronogeneous matrices, as well as the perspective division operator. The (5×5) identity matrix is denoted by $\mathcal{I}_5$. Similarly, the (3×3) identity matrix is denoted by $\mathcal{I}_3$.

The following matrix, $\mathcal{T}$, leaves the spatial location of a point unaltered, but advances the time component by one interframe time interval:

$$\mathcal{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The following equation holds:

$$\mathcal{T}\begin{pmatrix} x \\ y \\ z \\ t \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ t + \Delta T \\ 1 \end{pmatrix} \tag{4}$$

We now define the perspective projection matrix, $\mathcal{P}$. In our work, we use *left-handed* coordinate systems. For the camera coordinate system, the z-axis points in such a way that positive distances are in front of the camera. For simplicity, and without loss of generality, we assume that *all* distances are measured in the same units (this includes image plane coordinates). Then, with the camera coordinate system centered on the camera lens center,

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 & 0 \end{bmatrix} \tag{5}$$

where f is the focal length of the camera. Alternatively, if the camera coordinate system is centered on the image plane, then

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 & -1 \end{bmatrix} \tag{6}$$

The following operator, $\mathcal{D}$, is used in conjunction with $\mathcal{P}$ to define the image plane coordinates of a point in terms of its camera chronogeneous coordinates. The operator $\mathcal{D}$ is defined as

$$\mathcal{D}\left[\begin{pmatrix} x \\ y \\ z \\ t \\ \omega \end{pmatrix}\right] = \begin{pmatrix} x/\omega \\ y/\omega \end{pmatrix} \tag{7}$$

3.4.1 Coordinate Notations

Consider an image sequence, taken by a moving camera, consisting of "n_f" images (0 through $n_f - 1$), and consider a moving object having "n_p" points (0 through $n_p - 1$)

that are visible in each of these images. We assume that both ΔT, the interframe time interval, and f, the focal length of the camera, are known constants. Let $\vec{Q}_{i,j}$ refer to the position of the j^{th} point at the i^{th} discrete instant in time. We add a superscript to indicate in which reference frame and in what type of coordinates the position of the point is expressed.

$\vec{Q}^c_{i,j}$ is a point expressed in chronogeneous coordinates in the camera reference frame.

$\vec{Q}^s_{i,j}$ is the spatial (3D) part of the point $\vec{Q}^c_{i,j}$.

$\vec{Q}^p_{i,j}$ is a point expressed in image plane 2D coordinates. It is the exact projection of the point $\vec{Q}^c_{i,j}$ onto the image plane.

$\vec{Q}^m_{i,j}$ is the actually measured location of the point $\vec{Q}^p_{i,j}$ (in image plane 2D coordinates). It includes any "correspondence noise".

The following relationship expresses exact image plane coordinates in terms of camera chronogeneous coordinates:

$$\vec{Q}^p_{i,j} = \mathcal{D}[\mathcal{P}\vec{Q}^c_{i,j}] \tag{8}$$

where $\mathcal{P}$ is the perspective projection matrix, and $\mathcal{D}$ is the perspective division operator.

Assume that both the object and the camera are undergoing uniform chronogeneous motion. Let $\mathcal{A}$ be the (rigid) chronogeneous matrix that describes the motion of the camera. It describes a new camera position relative to the current instantaneous camera position. Let $\mathcal{B}$ be the chronogeneous matrix describing the motion of the object. Note that $\delta t_{\mathcal{A}} = \delta t_{\mathcal{B}} = \Delta T$.

3.5 Derivation of the "Coordinate Transformation Vector Equation"

The purpose of this section is to derive the coordinate transformation vector equation. This equation expresses the current chronogeneous position of a point in terms of its initial position, and the matrices describing the motion of the camera and the motion of the object. We first consider two subclasses of motion, and then derive the general coordinate transformation vector equation.

3.5.1 The case of a camera moving through a static environment

Consider a camera undergoing chronogeneous motion through a static environment. Motion of the camera has an inverse effect on object coordinates. Let us be more specific. Let us introduce the chronogeneous matrix $\mathcal{A}_R$ such that $\mathcal{A} = \mathcal{T}\mathcal{A}_R$, that is $\mathcal{A}_R = \mathcal{T}^{-1}\mathcal{A}$. The matrix $\mathcal{T}$ was defined previously, and causes time to advance by one "tick". The matrix $\mathcal{A}_R$ represents the spatial transformation that takes place between successive positions of the camera. Then the following relationships hold:

$$\mathcal{A}_R(\mathcal{T}^{-1}\vec{Q}^c_{i+1,j}) = \vec{Q}^c_{i,j}$$

and

$$\vec{Q}^c_{i+1,j} = \mathcal{T}\mathcal{A}_R{}^{-1}\vec{Q}^c_{i,j} = \mathcal{T}(\mathcal{T}^{-1}\mathcal{A})^{-1}\vec{Q}^c_{i,j} = \mathcal{T}\mathcal{A}^{-1}\mathcal{T}\vec{Q}^c_{i,j}$$

and therefore

$$\vec{Q}^c_{i,j} = (\mathcal{T}\mathcal{A}^{-1}\mathcal{T})^i\vec{Q}^c_{0,j} \tag{9}$$

If the camera is stationary, then $\mathcal{A}_R = \mathcal{I}_8$, $\mathcal{A} = \mathcal{T}$, and $\vec{Q}^c_{i,j} = \mathcal{T}^i\vec{Q}^c_{0,j}$.

3.5.2 The case of a stationary camera viewing a moving object

Consider a stationary camera viewing an object that is undergoing constant chronogeneous motion. The new camera chronogeneous coordinates of a point on the object are simply obtained by multiplying the current coordinates by $\mathcal{B}$, that is:

$$\vec{Q}^c_{i+1,j} = \mathcal{B}\vec{Q}^c_{i,j}$$

and in general

$$\vec{Q}^c_{i,j} = \mathcal{B}^i\vec{Q}^c_{0,j} \tag{10}$$

If the object is stationary, then $\mathcal{B} = \mathcal{T}$, and $\vec{Q}^c_{i,j} = \mathcal{T}^i\vec{Q}^c_{0,j}$.

3.5.3 Simultaneous camera and object motion

After considering the previous two cases, derivation of the coordinate transformation vector equation is straightforward. If the camera were stationary, then the chronogeneous position corresponding to the i^{th} image of a point would be $\mathcal{B}^i\vec{Q}^c_{0,j}$. If we consider only the spatial transformation involved, then the new position of the point due to object motion is $\mathcal{T}^{-i}\mathcal{B}^i\vec{Q}^c_{0,j}$. However, this point is viewed by a camera that has undergone motion. Therefore the composite effect is given by:

$$\vec{Q}^c_{i,j} = (\mathcal{T}\mathcal{A}^{-1}\mathcal{T})^i\mathcal{T}^{-i}\mathcal{B}^i\vec{Q}^c_{0,j} \tag{11}$$

The matrices in equation (11), the coordinate transformation vector equation, do not commute, and so the equation cannot be simplified. The implication of this vector equation is that, in general, simultaneous camera and object motion is not correctly modeled by either camera motion alone or object motion alone. This is due to the fact that, in general, there is no matrix, $\mathcal{C}$, such that $\mathcal{C}^i = (\mathcal{T}\mathcal{A}^{-1}\mathcal{T})^i\mathcal{T}^{-i}\mathcal{B}^i$. However, for every pure camera motion there is a pure object motion that has an identical effect on coordinate positions, and vice versa.

3.6 A Characterization of Constant Chronogeneous Motion

This section gives a characterization of the classes of motion that are representable by constant coefficient standard chronogeneous matrices. The following subsections discuss specific subcases in more detail. In each case we show how the components of the chronogeneous matrix are determined by the underlying parameters of motion, and how the motion parameters may be computed, given a chronogeneous matrix. At the end of this section, we briefly touch on the structure from chronogeneous motion problem.

Consider an arbitrary constant coefficient standard chronogeneous matrix, with deformation submatrix $\mathcal{S}$. This matrix represents the motion of some object, which is translating through space and structurally deforming according to the matrix $\mathcal{S}$. In addition, if the matrix expression $(\mathcal{I}_3 - \mathcal{S})$ is singular, the object may also be accelerating in a direction orthogonal to the subspace (of 3-space) spanned by $(\mathcal{I}_3 - \mathcal{S})$.

To make the foregoing more concrete, consider the case of a rigid object. In this case, the deformation submatrix $\mathcal{S}$ is a rotation matrix, call it $\mathcal{R}$. The matrix expression $(\mathcal{I}_3 - \mathcal{R})$ is singular. This is easily seen as follows. Let $\vec{A}$ be the (unit length) axis of rotation vector associated with $\mathcal{R}$. Then

$$(\mathcal{I}_3 - \mathcal{R})\vec{A} = \vec{A} - \mathcal{R}\vec{A} = \vec{A} - \vec{A} = \vec{0}$$

As the null space of $(\mathcal{I}_3 - \mathcal{R})$ contains a nonzero vector, this matrix expression is singular. Although we do not prove it here, $(\mathcal{I}_3 - \mathcal{R})$ is actually of rank 2, unless $\mathcal{R} = \mathcal{I}_3$. Therefore, if $\mathcal{R} \neq \mathcal{I}_3$, $\vec{A}$ spans the nullspace of $(\mathcal{I}_3 - \mathcal{R})$, and the general case of rigid chronogeneous motion corresponds to a rigid object rotating with fixed angular velocity, translating through space, and accelerating in the direction of the axis of rotation.

Figure 3-1 gives a taxonomy of the classes of rigid chronogeneous motion. These are discussed in the remainder of this section, after a discussion of the case of general deformation, with $(\mathcal{I}_3 - \mathcal{S})$ nonsingular.

3.6.1 A translating deforming object

Consider an object undergoing "constant deformation" about a center of deformation that is undergoing pure translation. Let $\vec{c}_i$ be the position of the center of deformation

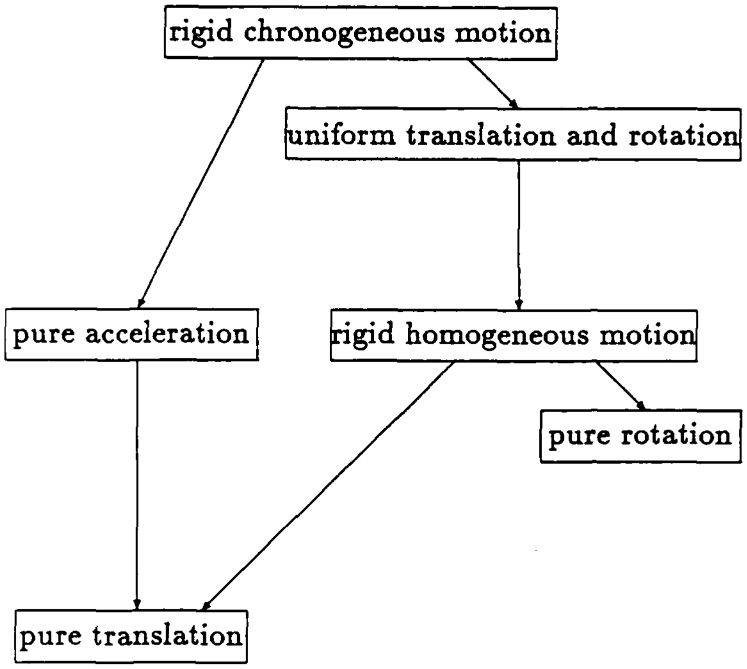


Figure 3-1: Taxonomy of Rigid Chronogeneous Motion

at the i^{th} discrete instant in time. Then the following equation recursively determines the 3D position of a given point on the object, for a given deformation matrix S:

$$(\vec{Q}^s_{i+1,j} - \vec{c}_{i+1}) = \mathcal{S}(\vec{Q}^s_{i,j} - \vec{c}_i)$$

For a translating object, $\vec{c}_i = \vec{c}_0 + t\vec{V} = \vec{c}_0 + i\vec{V}\Delta T$, for some initial center of deformation, $\vec{c}_0$, and velocity vector, $\vec{V}$. The above equation may then be rewritten as follows:

$$\begin{aligned}
\vec{Q}^s_{i+1,j} &= \mathcal{S}(\vec{Q}^s_{i,j} - \vec{c}_i) + \vec{c}_{i+1} \\
&= \mathcal{S}(\vec{Q}^s_{i,j} - (\vec{c}_0 + i\vec{V}\Delta T)) + (\vec{c}_0 + (i+1)\vec{V}\Delta T) \\
&= \mathcal{S}\vec{Q}^s_{i,j} - \mathcal{S}\vec{c}_0 - i\mathcal{S}\vec{V}\Delta T + \vec{c}_0 + i\vec{V}\Delta T + \vec{V}\Delta T \\
&= \mathcal{S}\vec{Q}^s_{i,j} + i(\mathcal{I}_3 - \mathcal{S})\vec{V}\Delta T + (\mathcal{I}_3 - \mathcal{S})\vec{c}_0 + \vec{V}\Delta T \\
&= \mathcal{S}\vec{Q}^s_{i,j} + t(\mathcal{I}_3 - \mathcal{S})\vec{V} + (\mathcal{I}_3 - \mathcal{S})\vec{c}_0 + \vec{V}\Delta T
\end{aligned}$$

The chronogeneous matrix representing the same motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c}
\multicolumn{3}{c|}{\mathcal{S}} & (\mathcal{I}_3 - \mathcal{S})\vec{V} & (\mathcal{I}_3 - \mathcal{S})\vec{c}_0 + \vec{V}\Delta T \\
\hline
0 & 0 & 0 & 1 & \Delta T \\
\hline
0 & 0 & 0 & 0 & 1
\end{array}\right] \qquad (12)$$

We assume $(\mathcal{I}_3 - \mathcal{S})$ is nonsingular. Then given an arbitrary matrix

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{S}} & \vec{\Gamma} & \vec{P} \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

we may compute the parameters of motion, $\vec{V}$ and $\vec{c}_0$, as follows:

$$\begin{aligned} \vec{V} &= (\mathcal{I}_3 - \mathcal{S})^{-1}\vec{\Gamma} & (13) \\ \vec{c}_0 &= (\mathcal{I}_3 - \mathcal{S})^{-1}(\vec{P} - \vec{V}\Delta T) & (14) \end{aligned}$$

3.6.2 Rigid chronogeneous motion: general case

Consider the case where the deformation is actually a rigid rotation, $\mathcal{R}$. Assume $\mathcal{R} \neq \mathcal{I}_3$. The case $\mathcal{R} = \mathcal{I}_3$ corresponds to pure acceleration, and is treated elsewhere.

As discussed in the introductory remarks to this section, the general case of rigid chronogeneous motion corresponds to a rigid object rotating with fixed angular velocity, translating through space, and accelerating in the direction of the axis of rotation. The following recursive relationship holds

$$(\vec{Q}^s_{i+1,j} - \vec{c}_{i+1}) = \mathcal{R}(\vec{Q}^s_{i,j} - \vec{c}_i),$$

where we may write

$$\vec{c}_i = \vec{c}_0 + t\vec{V} - \tfrac{1}{2}\gamma t^2 \vec{A} = \vec{c}_0 + i\vec{V}\Delta T - \tfrac{1}{2}i^2\gamma\vec{A}(\Delta T)^2$$

for some initial center of rotation, $\vec{c}_0$, initial velocity vector, $\vec{V}$, and signed magnitude of acceleration, γ. The axis of rotation, $\vec{A}$, is determined by $\mathcal{R}$. Substituting the formula for the position of the center of rotation into the recursive relationship, and simplifying, we obtain:

$$\begin{aligned} \vec{Q}^s_{i+1,j} &= \mathcal{R}\vec{Q}^s_{i,j} + t((\mathcal{I}_3 - \mathcal{R})\vec{V} - \gamma\vec{A}\Delta T) \\ &\quad + (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2 \end{aligned}$$

The chronogeneous matrix representing the same motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & \begin{array}{c}(\mathcal{I}_3 - \mathcal{R})\vec{V} \\ -\gamma\vec{A}\Delta T\end{array} & \begin{array}{c}(\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T \\ -\frac{1}{2}\gamma\vec{A}(\Delta T)^2\end{array} \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \qquad (15)$$

We now derive expressions for the motion parameters, $\vec{c}_0$, $\vec{V}$, and γ, in terms of $\mathcal{R}$, $\vec{A}$, $\vec{\Gamma}$, $\vec{P}$, and ΔT. The following relationships express the vector components of $\mathcal{B}$ in terms of the motion parameters:

$$\begin{aligned} \vec{\Gamma} &= (\mathcal{I}_3 - \mathcal{R})\vec{V} - \gamma\vec{A}\Delta T &(16)\\ \vec{P} &= (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2 &(17)\end{aligned}$$

We first derive an expression for γ. From equation (16):

$$\begin{aligned} \vec{A}\cdot\vec{\Gamma} &= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{V} - \gamma\vec{A}\Delta T)\\ &= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{V}) - \gamma(\vec{A}\cdot\vec{A})\Delta T\\ &= 0 - \gamma\Delta T\\ &= -\gamma\Delta T\end{aligned}$$

and therefore

$$\gamma = -(\vec{A}\cdot\vec{\Gamma})/\Delta T$$

In the above, the symbol "·" indicates dot product.

Next, we derive an expression for $\vec{V}$. From equation (17):

$$\begin{aligned} \vec{A}\cdot\vec{P} &= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2)\\ &= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{c}_0) + \vec{A}\cdot(\vec{V}\Delta T) - \tfrac{1}{2}\gamma(\vec{A}\cdot\vec{A})(\Delta T)^2\\ &= 0 + (\vec{A}\cdot\vec{V})\Delta T - \tfrac{1}{2}(-(\vec{A}\cdot\vec{\Gamma})/\Delta T)(\Delta T)^2\\ &= (\vec{A}\cdot\vec{V})\Delta T + \tfrac{1}{2}(\vec{A}\cdot\vec{\Gamma})\Delta T\end{aligned}$$

and therefore

$$\begin{aligned} \vec{A}\cdot\vec{V} &= ((\vec{A}\cdot\vec{P}) - \tfrac{1}{2}(\vec{A}\cdot\vec{\Gamma})\Delta T)/\Delta T\\ &= \vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma})\end{aligned}$$

In the following, the symbol "+" used as an exponent denotes the pseudoinverse operation. Readers who are unfamiliar with the pseudoinverse are referred to [10]. The pseudoinverse is a generalization of the inverse that also applies when the matrix is not square, or not of full rank (as in the following). We make use of the fact that $(\mathcal{I}_3 - \mathcal{R})^+\vec{A} = \vec{0}$. The initial velocity vector, $\vec{V}$, is determined as follows:

$$\begin{aligned} \vec{V} &= (\mathcal{I}_3 - \mathcal{R})^+(\mathcal{I}_3 - \mathcal{R})\vec{V} + (\vec{A}\cdot\vec{V})\vec{A}\\ &= (\mathcal{I}_3 - \mathcal{R})^+(\vec{\Gamma} + \gamma\vec{A}\Delta T) + (\vec{A}\cdot\vec{V})\vec{A}\\ &= (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + \gamma(\mathcal{I}_3 - \mathcal{R})^+\vec{A}\Delta T + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A}\\ &= (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A}\end{aligned}$$

Finally, we derive an expression for $\vec{c}_0$. The initial center of rotation, $\vec{c}_0$, is not uniquely determined. Adding any real multiple of $\vec{A}$ to $\vec{c}_0$ results in a physically indistinguishable motion. The derived value of $\vec{c}_0$ is the minimum length solution. From equation (17):

$$\begin{aligned}
\vec{c}_0 &= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - \vec{V}\Delta T + \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2) \\
&= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - \vec{V}\Delta T) \\
&= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - ((\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A})\Delta T) \\
&= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma}\Delta T - (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A}\Delta T) \\
&= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma}\Delta T)
\end{aligned}$$

In summary, the motion parameters may be computed as follows:

$$\gamma = -(\vec{A}\cdot\vec{\Gamma})/\Delta T \tag{18}$$

$$\vec{V} = (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A} \tag{19}$$

$$\vec{c}_0 = (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma}\Delta T) \tag{20}$$

3.6.3 Uniform translation and rotation

For this subclass of rigid chronogeneous motion, $\gamma = 0$. This class of motion has eight degrees of freedom. There are three degrees of rotational freedom, three degrees of freedom for the velocity vector, and two degrees of freedom for the center of rotation. The chronogeneous matrix representing this motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & (\mathcal{I}_3 - \mathcal{R})\vec{V} & (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{21}$$

3.7 Rigid Homogeneous Motion

This case corresponds to the class of motion representable by homogeneous transformations. $\vec{\Gamma} = \vec{0}$ in this case, and this class of motion has six degrees of freedom. For $\mathcal{R} \neq \mathcal{I}_3$, a general motion of this class consists of rotation, coupled with a restricted form of translation. Translation, if any, occurs in the direction of the axis of rotation. This may more commonly be described as helical or "barberpole" motion. The following recursive relationship holds

$$(\vec{Q}^{s}_{i+1,j} - \vec{c}_{i+1}) = \mathcal{R}(\vec{Q}^{s}_{i,j} - \vec{c}_i)$$

where

$$\vec{c}_i = \vec{c}_0 + t\nu\vec{A} = \vec{c}_0 + i\nu\vec{A}\Delta T$$

for some initial center of rotation, $\vec{c}_0$, and (signed) magnitude of velocity, ν. The chronogeneous matrix representing this motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & \vec{0} & (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \nu\vec{A}\Delta T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{22}$$

Given a chronogeneous matrix of the above form, the motion parameters, ν and $\vec{c}_0$, may be computed as follows:

$$\nu = (\vec{A}\cdot\vec{P})/\Delta T \tag{23}$$

$$\vec{c}_0 = (\mathcal{I}_3 - \mathcal{R})^+\vec{P} \tag{24}$$

3.7.1 Pure rotation

Pure rotation is a subclass of rigid homogeneous motion, with $\nu = 0$. This class of motion has five degrees of freedom. There are three degrees of rotational freedom, and two degrees of freedom for the center of rotation. The recursive relationship simplifies to

$$(\vec{Q}^s_{i+1,j} - \vec{c}_0) = \mathcal{R}(\vec{Q}^s_{i,j} - \vec{c}_0)$$

and the chronogeneous matrix corresponding to this motion is

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & \vec{0} & (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{25}$$

3.7.2 Pure acceleration

For this subclass of rigid chronogeneous motion, $\mathcal{R} = \mathcal{I}_3$. This class of motion has six degrees of freedom. The following relationships hold:

$$\begin{aligned} \vec{Q}^s_{i,j} &= \vec{Q}^s_{0,j} + t\vec{V} - \tfrac{1}{2}\gamma t^2\vec{A} \\ &= \vec{Q}^s_{0,j} + i\vec{V}\Delta T - \tfrac{1}{2}i^2\gamma\vec{A}(\Delta T)^2 \end{aligned}$$

for some initial velocity vector, $\vec{V}$, magnitude of acceleration, γ, and axis of *acceleration*, $\vec{A}$. When comparing this subclass of motion to the general case, we see that there is no reference to a center of rotation, and the axis of rotation has been replaced by an axis of acceleration, that is free to point in any direction. The recursive relationship for this motion is

$$\vec{Q}^{\bullet}_{i+1,j} = \vec{Q}^{\bullet}_{i,j} - t\gamma\vec{A}\Delta T + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2$$

and the chronogeneous matrix representing this motion is

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & -\gamma\vec{A}\Delta T & \vec{V}\Delta T - \frac{1}{2}\gamma\vec{A}(\Delta T)^2 \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{26}$$

Given a chronogeneous matrix, with $\mathcal{S} = \mathcal{R} = \mathcal{I}_3$, the motion parameters, γ, $\vec{A}$, and $\vec{V}$, may be computed as follows:

$$\gamma = \|\vec{\Gamma}\|/\Delta T \tag{27}$$

$$\vec{A} = -\vec{\Gamma}/\|\vec{\Gamma}\| \tag{28}$$

$$\vec{V} = \vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma} \tag{29}$$

The signs of equations (27) and (28) are chosen to be consistent with equation (18), and so that γ is nonnegative.

3.7.3 Pure translation

For pure translation, $\mathcal{R} = \mathcal{I}_3$, and $\vec{\Gamma} = \vec{0}$. This class of motion has three degrees of freedom. The recursive relationship simplifies to

$$\vec{Q}^{\bullet}_{i+1,j} = \vec{Q}^{\bullet}_{i,j} + \vec{V}\Delta T$$

which implies

$$\vec{Q}^{\bullet}_{i,j} = \vec{Q}^{\bullet}_{0,j} + t\vec{V} = \vec{Q}^{\bullet}_{0,j} + i\vec{V}\Delta T$$

The chronogeneous matrix corresponding to this motion is

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & \vec{0} & \vec{V}\Delta T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{30}$$

and the following relationship expresses $\vec{V}$ in terms of the $\vec{P}$ subvector of $\mathcal{B}$:

$$\vec{V} = \vec{P}/\Delta T \tag{31}$$

3.7.4 Structure from chronogeneous motion

The author is currently working on solving the structure from motion problem for an object undergoing constant chronogeneous motion. Details of the solutions will be presented in a future paper. Table 1 summarizes the number of frames required to solve this problem for a given number of points, for both rigid and nonrigid motion.

Rigid Chronogeneous Motion	
Points	Frames
1	6
2	4
3	3

Nonrigid Chronogeneous Motion	
Points	Frames
1	9
2	5
3	4
5	3

Table 1: Number of Frames Required to Solve SFM Problem

3.8 Recovery of Absolute Depth from a Monocular Image Sequence

In this section, we present a novel application of the methodology developed in this paper. We show how, under certain circumstances, absolute depth may be recovered from a monocular image sequence.

Assume that a (rigid) object, undergoing constant chronogeneous motion, is imaged by a stationary camera (perspective projection). Let us ignore any measurement or correspondence error, and assume that the structure from motion problem has been solved for chronogeneous motion. (We will present a solution to this problem in an upcoming paper). Let

$$\mathcal{B}_T = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}_T} & \vec{\Gamma}_T & \vec{P}_T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

be the "true" matrix representing the motion of the object, and let

$$\mathcal{B}_S = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}_S} & \vec{\Gamma}_S & \vec{P}_S \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

be the computed solution to the structure from motion problem. Then the following relationships hold

$$\mathcal{R}_T = \mathcal{R}_S \qquad (32)$$

$$\vec{\Gamma}_T = \lambda\vec{\Gamma}_S \tag{33}$$
$$\vec{P}_T = \lambda\vec{P}_S \tag{34}$$

where $\lambda > 0$ is an unknown scale factor. The above relationships reflect the fact that, without additional assumptions, the depth can only be determined to within an unknown (positive) scale factor.

We now make the additional assumption that the object is accelerating solely due to a constant external force of known magnitude, and show how the scale factor may be recovered. This, in turn, allows the true chronogeneous matrix, and hence the absolute parameters of motion, and the absolute distances to points on the object to be recovered. This assumption is reasonable for certain objects, such as a falling apple, or a cannonball (neglecting air resistance). Such objects are undergoing "ballistic" motion. The object may be rotating, but in order for the motion to be chronogeneous, the direction of the axis of rotation must be aligned with the direction of the external force (gravity in this case). In other words, the axis of rotation must point either "upward" or "downward", with respect to the force vector. In order to determine the scale factor, we use the fact that the *magnitude* of the acceleration is the same in all inertial reference frames. In the following, let g be the acceleration due to the external force.

For the general case of rigid chronogeneous motion, we have from equation (18):

$$\begin{aligned} g &= |\gamma_T| \\ &= |-(\vec{A}_T\cdot\vec{\Gamma}_T)/\Delta T| \\ &= |-(\vec{A}_S\cdot(\lambda\vec{\Gamma}_S))/\Delta T| \\ &= |\lambda(\vec{A}_S\cdot\vec{\Gamma}_S)/\Delta T| \end{aligned}$$

and therefore

$$\lambda = |g\Delta T/(\vec{A}_S\cdot\vec{\Gamma}_S)| = g\Delta T/|\vec{A}_S\cdot\vec{\Gamma}_S| \tag{35}$$

For the subcase of pure acceleration, the above equation holds if we identify the axis of rotation, and the axis of acceleration. However, a more direct derivation is possible. From equation (27):

$$g = |\gamma_T| = \|\vec{\Gamma}_T\|/\Delta T = \|\lambda\vec{\Gamma}_S\|/\Delta T = \lambda\|\vec{\Gamma}_S\|/\Delta T$$

and therefore

$$\lambda = g\Delta T/\|\vec{\Gamma}_S\| \tag{36}$$

3.9 Conclusion and Future Research

In this section, we present what we see as the major contributions of this research. In addition, we discuss related current and future research of the author.

The first contribution of this research is the *general* nature of the representation. A fairly large and interesting class of motion may be represented. Rotation, translation, and fixed axis motion, as well as (possibly restricted forms of) acceleration are all representable. Furthermore, the represention of rigid and nonrigid motion is unified.

Chronogeneous transformation matrices also provide a *compact representation* of a fairly large class of camera/object motion, and allow the *efficient computation* of the motion of computer generated objects. It is straightforward to calculate a chronogeneous matrix given the underlying motion parameters, and vice versa. Chronogeneous coordinates should thus prove very useful in the fields of computer graphics and animation.

Next, this research *unifies the representation of camera and object motion.* The coordinate transformation vector equation provides the connection between the two. Previous researchers have studied problems involving either camera motion, or object motion, but not both simultaneously. Sometimes the distinction between the two has been ignored. This is mainly because so much research has been devoted to the analysis of the two frames case, where camera and object motion are confounded. It is only when at least three frames are available that these two motions can, to a large extent, (locally) be disambiguated.

Finally, this representation *models physically natural motion.* The importance of this fact is that, by taking advantage of the constraints imposed by the spatio-temporal continuity of such motion, we may be able to (and for chronogeneous motion are able to) solve the structure from motion problem using fewer points and/or frames than when only rigidity is imposed. Furthermore, *structure from nonrigid motion* may also be studied.

The author is currently working on solving the structure from motion problem for an object undergoing constant chronogeneous motion. Details of the solutions will be presented in a future paper.

Also, the coordinate transformation vector equation expresses the motion of a point in terms of the camera chronogeneous matrix, and the object chronogeneous matrix. Solutions to this equation will allow the simultaneous recovery of camera and object motion, to within certain inherent ambiguities.

Finally, as a further research problem, it should be possible to estimate the coefficients of the chronogeneous matrix using Kalman filtering techniques. The parameters of motion could then be determined using the equations developed in this paper.

A Useful Formulae Involving Standard Chronogeneous Matrices

In this appendix, we derive formulae for the product of two standard chronogeneous matrices, the inverse of a standard chronogeneous matrix, and the powers of a standard chronogeneous matrix. Standard chronogeneous matrices are closed under each of these operations, and therefore form a group under matrix multiplication.

A.1 Formulae for Matrix Products

Let

$$\mathcal{C}_A = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}_A} & \vec{\Gamma}_A & \vec{P}_A \\ \hline 0 & 0 & 0 & 1 & \delta t_A \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and let

$$\mathcal{C}_B = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}_B} & \vec{\Gamma}_B & \vec{P}_B \\ \hline 0 & 0 & 0 & 1 & \delta t_B \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Then

$$\mathcal{C}_A\mathcal{C}_B = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}_A\mathcal{S}_B} & \mathcal{S}_A\vec{\Gamma}_B + \vec{\Gamma}_A & \mathcal{S}_A\vec{P}_B + \vec{\Gamma}_A\delta t_B + \vec{P}_A \\ \hline 0 & 0 & 0 & 1 & \delta t_B + \delta t_A \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and

$$\mathcal{C}_B\mathcal{C}_A = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}_B\mathcal{S}_A} & \mathcal{S}_B\vec{\Gamma}_A + \vec{\Gamma}_B & \mathcal{S}_B\vec{P}_A + \vec{\Gamma}_B\delta t_A + \vec{P}_B \\ \hline 0 & 0 & 0 & 1 & \delta t_A + \delta t_B \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Standard chronogeneous matrices are, in general, not commutative.

For the special case of the matrix $\mathcal{T}$,

$$\mathcal{T}\mathcal{C} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}} & \vec{\Gamma} & \vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t + \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and

$$\mathcal{C}\mathcal{T} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}} & \vec{\Gamma} & \vec{\Gamma}\Delta T + \vec{P} \\ \hline 0 & 0 & 0 & 1 & \Delta T + \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The matrix $\mathcal{T}$ does not, in general, commute with other standard chronogeneous matrices. However, it does commute if $\vec{\Gamma} = \vec{0}$.

A.2 Formulae for the Inverse of a Matrix

If the deformation submatrix, $\mathcal{S}$, of a standard chronogeneous matrix, $\mathcal{C}$, is invertible, then the inverse of $\mathcal{C}$ exists and

$$\mathcal{C}^{-1} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}^{-1}} & -\mathcal{S}^{-1}\vec{\Gamma} & -\mathcal{S}^{-1}(\vec{P}-\vec{\Gamma}\delta t) \\ \hline 0 & 0 & 0 & 1 & -\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The above formula may be verified by multiplying the matrix and its inverse. Remember that for a rotation matrix, the inverse of the matrix is simply its transpose. Therefore, when $\mathcal{S}$ is a rotation matrix, call it $\mathcal{R}$, the formula for the inverse becomes:

$$\mathcal{C}^{-1} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}^{T}} & -\mathcal{R}^{T}\vec{\Gamma} & -\mathcal{R}^{T}(\vec{P}-\vec{\Gamma}\delta t) \\ \hline 0 & 0 & 0 & 1 & -\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

If $\mathcal{S} = \mathcal{I}_3$, the formula simplifies to:

$$\mathcal{C}^{-1} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & -\vec{\Gamma} & -\vec{P}+\vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & -\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

As a special case, the inverse of the matrix $\mathcal{T}$ is given by

$$\mathcal{T}^{-1} = \left[\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\Delta T \\ 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

A.3 Formulae for Integer Powers of a Matrix

The formula for an arbitrary integer power (≥ 2) of a standard chronogeneous matrix is given below. This formula is easily proved by induction.

$$\begin{aligned} \mathcal{C}^{i} &= \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}_i} & \vec{\Gamma}_i & \vec{P}_i \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}^{i}} & (\sum_{k=0}^{i-1}\mathcal{S}^{k})\vec{\Gamma} & (\sum_{k=0}^{i-1}\mathcal{S}^{k})\vec{P} + (\sum_{k=0}^{i-2}(i-1-k)\mathcal{S}^{k})\vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{aligned}$$

where we use the convention that the 0th power of a (3×3) matrix is the (3×3) identity matrix, $\mathcal{I}_3$, and the following recurrence relations hold:

$$\begin{aligned} \mathcal{S}_{i+1} &= (\mathcal{S})\mathcal{S}_i \\ \vec{\Gamma}_{i+1} &= \mathcal{S}\vec{\Gamma}_i + \vec{\Gamma} \\ \vec{P}_{i+1} &= \mathcal{S}\vec{P}_i + \vec{\Gamma}(i\delta t) + \vec{P} \end{aligned}$$

If $\mathcal{S} = \mathcal{I}_3$, then the above formula simplifies to

$$\mathcal{C}^i = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & i\vec{\Gamma} & i\vec{P} + \frac{1}{2}i(i-1)\vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and holds for all $i \geq -1$. Here we use the convention that the 0th power of a (5×5) matrix is the (5×5) identity matrix, $\mathcal{I}_5$. The following analogous formula holds when $i \leq 0$:

$$\mathcal{C}^i = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & i\vec{\Gamma} & i\vec{P} - \frac{1}{2}i(i+3)\vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The formula for an arbitrary power of the matrix $\mathcal{T}$ is simply:

$$\mathcal{T}^i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & i\Delta T \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This formula holds for *all* integer powers.

A.4 Special Matrix Forms

Let $\mathcal{H}$ be the purely spatial and time independent rigid chronogeneous transformation defined as

$$\mathcal{H} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}} & \vec{0} & \vec{T} \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Then

$$\mathcal{H}^{-1} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}^T} & \vec{0} & -\mathcal{R}^T\vec{T} \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and the following relationships hold:

$$\mathcal{H}\mathcal{C} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}\mathcal{S}} & \mathcal{R}\vec{\Gamma} & \mathcal{R}\vec{P}+\vec{T} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{H}^{-1}\mathcal{C} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}^T\mathcal{S}} & \mathcal{R}^T\vec{\Gamma} & \mathcal{R}^T(\vec{P}-\vec{T}) \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{H}\mathcal{C}\mathcal{H}^{-1} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}\mathcal{S}\mathcal{R}^T} & \mathcal{R}\vec{\Gamma} & (\mathcal{I}_3-\mathcal{R}\mathcal{S}\mathcal{R}^T)\vec{T}+\mathcal{R}\vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{H}^{-1}\mathcal{C}\mathcal{H} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{R}^T\mathcal{S}\mathcal{R}} & \mathcal{R}^T\vec{\Gamma} & \mathcal{R}^T(\vec{P}+(\mathcal{S}-\mathcal{I}_3)\vec{T}) \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

If $\mathcal{S} = \mathcal{I}_3$, then the last two equations may be simplified as follows:

$$\mathcal{H}\mathcal{C}\mathcal{H}^{-1} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & \mathcal{R}\vec{\Gamma} & \mathcal{R}\vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{H}^{-1}\mathcal{C}\mathcal{H} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & \mathcal{R}^T\vec{\Gamma} & \mathcal{R}^T\vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The following form is useful when $\delta t_{\mathcal{C}} = \Delta T$:

$$\mathcal{T}\mathcal{C}^{-1}\mathcal{T} = \left[\begin{array}{c|c|c|c|c} \multicolumn{3}{c|}{\mathcal{S}^{-1}} & -\mathcal{S}^{-1}\vec{\Gamma} & -\mathcal{S}^{-1}\vec{P} \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

References

[1] Hans-Hellmut Nagel. Image sequences - ten (octal) years - from phenomenology towards a theoretical foundation. In *Proceedings of the Eighth International Conference on Pattern Recognition*, pages 1174–1185. IEEE Computer Society Press, October 27–31, 1986.

[2] Shimon Ullman. *The Interpretation of Visual Motion*. MIT Press, Cambridge, Massachusetts, 1979.

[3] Hormoz Shariat. *The Motion Problem: How to use more than two frames.* PhD thesis, IRIS, School of Engineering, University of Southern California, October 1986.

[4] Su shing Chen. Structure-from-motion without the rigidity assumption. In *Proceedings of the Third Workshop on Computer Vision: Representation and Control*, pages 105–112. IEEE Computer Society Press, October 13–16, 1985.

[5] David J. Heeger and Alex P. Pentland. Seeing structure through chaos. In *Proceedings of the Workshop on Motion: Representation and Analysis*, pages 131–136. IEEE Computer Society Press, May 7–9, 1986.

[6] Muralidhara Subbarao. Interpretation of image motion fields: A spatio-temporal approach. In *Proceedings of the Workshop on Motion: Representation and Analysis*, pages 157–165. IEEE Computer Society Press, May 7–9, 1986.

[7] Su shing Chen and Michael Penna. Recognizing deformations of nonrigid bodies. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 452–455. IEEE Computer Society Press, June 22–26, 1986.

[8] L. G. Roberts. Machine perception of three-dimensional solids. In J. T. Tippett et al., editors, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, Massachusetts, 1968.

[9] Richard P. Paul. *Robot Manipulators: Mathmatics, Programming, and Control*, chapter 1. MIT Press, Cambridge, Massachusetts, 1981. Chapter heading: Homogeneous Transformations.

[10] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems.* Prentice-Hall, Inc., Englewood Cliffs, N. J., 1974. See chapter 7 for basics of pseudoinverse.

4 SPATIO-TEMPORAL ANALYSIS OF AN IMAGE SEQUENCE WITH OCCLUSION

Motion understanding is one of the most important visual functions, and has numerous applications in robotics and industrial automation. The information extracted from this process include segmentation, range, velocity, and so on. Motion therefore plays a basic role in the understanding process. It seems very reasonable that animals have perceptual systems or subsystems purely based on motion [25]. Some animals are known to shake their heads to gather information for hunting. Visual processing is neither a pure bottom up processing nor a pure top down one. Communication and feedback are necessary between high level and low level processing. In this paper, we try to identify and simulate a low level, local mechanism for motion detection.

The method to perform motion detection and understanding introduced in this paper is basically domain independent. It is able to calculate flow from discrete images and to separate objects based on their motion alone. The procedure developed here is not computationally intensive, and the information needed is only local in nature, making a VLSI implementation possible [15].

The following assumptions and restrictions are made regarding the observed sequence of images [20] :

1. *Maximum velocity*
   the operator is only sensitive to a finite range of velocity. An object can move at most $V \cdot dt$ between two images taken dt time units apart,
2. *Small velocity change*
   it is a consequence of physical laws and the assumption of high sampling rate,
3. *Small shape change*
   each object is either rigid or is changing its shape slowly,
4. *Common motion*
   objects are spatially coherent and therefore appear in images as regions of points sharing a *common motion*,
5. *Causality*
   objects cannot appear or disappear suddenly.

The principle behind our approach is to find the velocity components of an edge point along several different directions and estimate its *normal* velocity, that is the velocity in the direction normal to the direction of the edge, subject to the constraints listed above.

The next section is a brief review of previous work in motion analysis, in which three different approaches are discussed and compared. In section 3, the basic idea of combining spatial and temporal information is introduced. The method we are proposing to segment objects from a sequence of images is discussed and formalized in the section as well. Several results are given in section 4 to illustrate how the method works. There are results on both synthetic and real image sequences. Finally, a summary of remarks is contained in section 5.

4.1 Previous Work

Motion analysis is a strong research area in computer vision. The key to understanding image sequences lies in the analysis of differences and similarities between consecutive time frames. The approaches taken differ in the type of primitives used for matching, the criteria used to resolve ambiguities and the number of frames in the sequences. There can be broadly classified as follows:

4.1.1 Feature-based approaches

This approach is probably the most intuitive if identifiable spatial features can be extracted and then the correspondences are possible to establish. A variety of possible features have been tried: points, line segments [17], blobs, local edges [12], vertices [2], local maxima of variability [3,18], local statistics [25], extrema of the local grey value curvature [7], corners[6,24], regions [21,27] or even recognized objects. Good features are those which can minimize the effect of illumination and geometric changes. The higher the level of descriptions at which matching is attempted, the less ambiguous the matching process will be, but this gain may be offset by the errors and deficiencies of the current programs producing those descriptions. The sampling rate may be large as long as the features are still present in the images. The accuracy is high if a sharp and localized feature is tracked, but such a desired feature may be hard to find.

The extracted features of images are then matched to calculate a set of disparity vectors for the sequence. The correspondence is established based on a metric affinity function as well as a group mapping criterion. The best match is found based on an optimization criterion. Criterion functions can range from simple cross-correlation [9] to sophisticated graph-matching procedures [12]. The matching process is computational expensive. Methods such as coarse-to-fine resolution matching [10] may be used to speed up the process.

4.1.2 Intensity-based approaches

This approach can be subdivided further in three:
The first type is a *differencing scheme* which is done by subtracting one image from the other and thresholding the result. The clusters of points in the difference image correspond to moving objects. By ignoring the stationary background, the computational resources are focused on the moving objects [14]. This scheme prefers large motion so that the interesting objects are far enough not to overlap in position in different images, because the interior of homogeneous regions do not generate a difference. It fails when the observer is moving or when the illumination is not constant.

The second type is a *correlation scheme.* A patch of the image is used as a template and cross-correlated with other images. The peak value indicates a match in intensity and defines a disparity for the image patch [13]. This scheme suffers from the following limitations [17]:

1. It requires the presence of a detectable texture within each correlation window, and therefore tends to fail in featureless or repetitive texture environment.
2. It tends to be confused by the presence of a surface discontinuity in a correlation window.
3. It is sensitive to absolute intensity, contrast, and illumination.
4. It gets confused in rapidly changing depth fields (e.g., vegetation).

The third type is a *gradient scheme* which is widely used for the calculation of optical flow [8]. If $I(x,y)$ denotes the intensity function of the image, then the following holds:

$$-\frac{\partial I}{\partial t} = G_x \cdot u + G_y \cdot v \tag{1}$$

where $\frac{\partial I}{\partial t}$ is the temporal intensity change at position (x,y); G_x and G_y represent the intensity gradient at the image point; and u, v are local velocities in the x and y directions, respectively. Since $\frac{\partial I}{\partial t}$, G_x and G_y are all measurable by the observer, u and v can be determined by the above relation.

Anandan suggested a framework to compute dense field of displacement vectors with associated confidence measures [1]. In general, intensity-based approaches are faster at the cost of high data volume. The images must be analyzed for every few pixels of displacement, which means a high sampling rate. This approach allows complex shape

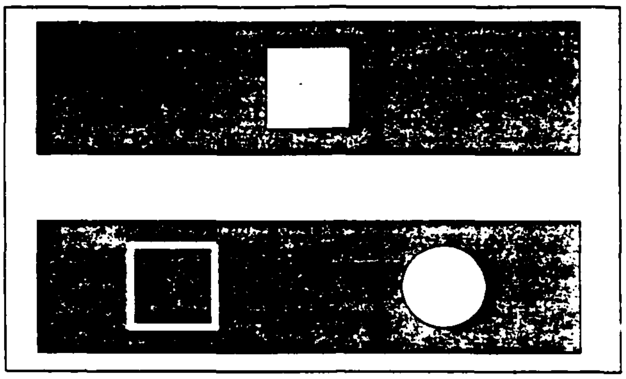


Figure 4-1: Features of Objects

changes and introduces the many-to-one match problem. It is also very noise sensitive and less accurate due to ambiguity of local measurements. A VLSI analog circuit was designed at Caltech to implement equation (1) [26]. The local ambiguity due to the *aperture problem* is handled by a constraint-solving circuit.

An interesting experiment demonstrates that the feature-based approach is a high level processing while the other one is low level [22], see fig. 4-1. A solid square is shown in the center against a dark background and is then replaced with an outline square on the left and a solid circle on the right.

The viewer who is confronted with these images usually sees the square moving toward the circle rather than toward the outlined square, but when the images are presented slowly and there is time to scrutinize the image, then the perception is that the square moves to the outline square. This suggests that regions of low spatial frequencies (smooth intensity change) are more likely to be detected initially, which would suggest that intensity processing is performed by a preprocessor.

4.1.3 Image-sequence-based approach

There is still another approach using a sequence of closely spaced images. This approach has received little attention until recently because of the huge amount of storage and computation involved. A solid of data called *spatio-temporal data*, with time as the third dimension, was introduced by Bolles and Baker [4]. It is constructed by a sequence of images close enough that none of the objects moves more than a pixel or so between frames. The *epipolar-plane image*, or EPI, is a slice taken from the spatio-temporal data along the temporal dimension. They used EPI to simplify the matching phase in stereo analysis. Consider a simple lateral motion in which a camera moves from right to left along a straight track and takes pictures at constant distance with its optical axis orthogonal to its direction of motion. Any feature point P describes a linear trajectory on the EPI because the only motion is horizontal and constant. The slope of the line determines the distance from the point to the camera. Occlusion is also immediately apparent in this representation. Those linear trajectories are then extracted by a non-directional Laplacian-Gaussian filter which treats the time domain the same as the horizontally spatial domain. Therefore the edge features are mixed with the intensity discontinuities due to occlusions. An extended work using projective duality is proposed in [16], which is expected to generalize the linear camera motion to an arbitrary one. This will still be applicable only if the camera path is known, and if the scene is frozen. To generalize this idea for motion analysis, consider the case where the camera is fixed. The motion of an image point still gives a continuous trajectory in the spatio-temporal data but it is not necessarily to be a straight line and, in general, does not fall on any EPI. A new approach is to be discussed in the next section to recover the trajectory called a *path* in order to derive the motion information. In contrast to some previous methods which require the acquisition of the complete spatio-temporal volume before processing is done, the method described here provides estimation after a few frames, and refining them as more frames come in. It therfore makes better use of storage and processing is faster.

4.2 Description of the Approach

From many biological experimental evidences, the primitive animal visual processing can be modeled as a nonlinear system which is a function of time and space. The system function is basically a composition of a spatial bandpass filter and a temporal bandpass filter. The central frequency and bandwidth define the range and sensitivity of its motion detection ability. The filtering effect permits to find the highest correlations in both temporal and spatial domain.

The goal of this paper is trying to devise a primitive parallel process which is able to extract motion information locally from the intensity image. The extracted information is passed to the higher level for a globally consistent interpretation.

4.2.1 Basic Idea

In many low level biological visual systems, edges are always one of the most useful features detected by the front-end preprocessing. When we look at a scene with moving objects, we are first alerted by the moving edges and then the movements propagate into the interior of the corresponding regions. At this moment, our internal representation of the scene becomes a bunch of surface patches associated with velocities. Those surfaces may be matched with our internal models to recognize moving objects. Motion is not the

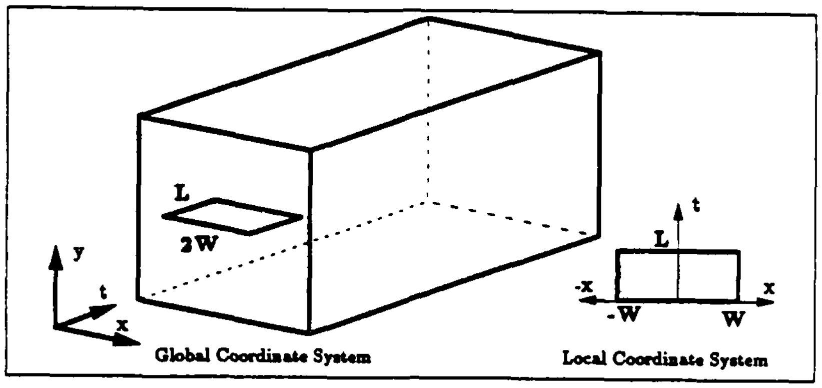


Figure 4-2: A *slice* from the sequence

only cue humans use to visualize the world, but some other life forms do rely on motion exclusively, e.g. the predacious activity of the frog. They prey only on moving worms or insects and their attention is never attracted by stationary objects.

The motion information we want to extract is the normal flow associated with the edge elements. The aperture effect restricts us so that only one component of the motion in the 2-D image can be estimated.

Assuming a dense image sequence is available, the method chosen for the normal flow estimation is basically a spatio-temporal analysis on the *slices* constructed from the image sequence. A *slice* is a collection of **L** 1-D images of width **2W** taken from **L** successive frames in the sequence at the same position, see fig. 4-2. It can be displayed as an image, the vertical and horizontal axes corresponding to the time and spatial directions respectively.

This spatio-temporal data structure provides an easy way to trace a line segment through frames. Assume there is an edgel **P** on a line segment under translation $\vec{V}$ in frame i, it moves to **P**$'$ in frame j. If we construct a slice centered at **P** with inclination θ, the 1-D image in the jth frame picks up another point **P**$''$ because in general the orientation of the slice is different from that of the translation $\vec{V}$, see fig. 4-3.

Since we assume high sampling rate, there are points between **P** and **P**$''$ corresponding to the line segment in frames in between, the sequence of those points is called the *path* of **P** in the *slice* . The slope of the path gives an estimate of the speed form **P** to **P**$''$,

$$V_\theta = \frac{\mathbf{P}'' - \mathbf{P}}{t_j - t_i} = \frac{\Delta x}{\Delta t}$$

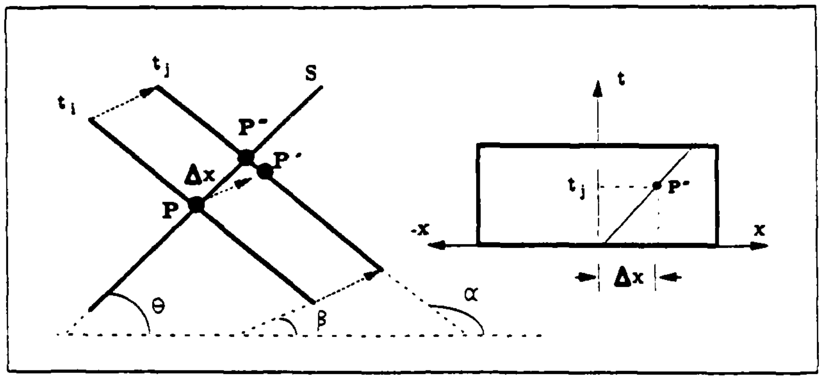


Figure 4-3: A *path* in a *slice*

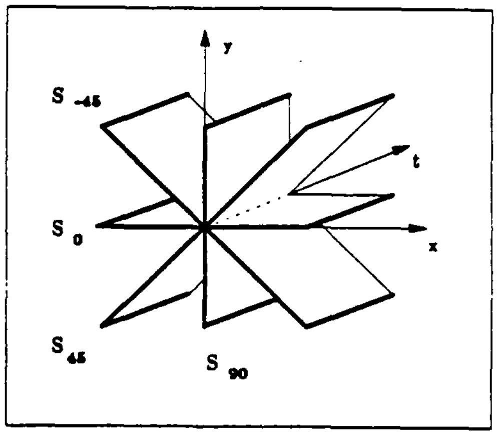


Figure 4-4: Different orientations of *slices*

The longer the path can be traced, the better the estimate of V_θ. Therefore all the V_θ estimates are associated with a *confidence factor* proportional to the length of the corresponding path.

V_θ alone is not enough to determine the real velocity $\vec{V}$, it only provides a constraint that the projection of $\vec{V}$ onto the normal of the normal of the line segment should be the same as the component of V_θ along the normal direction. Let α be the inclination of the line segment and β be the orientation of $\vec{V}$, then we have the following relation

$$V_\theta = \frac{\|\vec{V}\| \cdot \sin(\alpha - \beta)}{\sin(\alpha - \theta)}$$

The orientation of the 1-D image, θ, may be arbitrary, we choose the most convenient

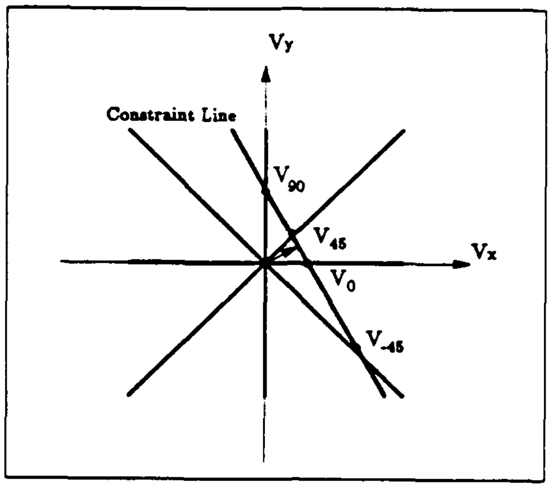


Figure 4-5: The normal velocity and *constraint line* in velocity space

four orientations: $-45°$, $0°$, $45°$ and $90°$. The corresponding *slices* are called S_{-45}, S_0, S_{45} and S_{90}, see fig. 4-4.

For each edge point detected by the Canny edge detector [5], four *slices* are constructed with all the 1-D images form frame 0 to frame **L - 1** centered at the position of the edgel in frame 0. The velocities estimated from the *slices* fall on a line in the velocity space, see fig. 4-5. We can simply fit a line to the velocity points based on least square error weighted by the *confidence factor*, and find the perpendicular vector from origin to the line. The perpendicular vector is the normal velocity $\vec{N}$ and the fitted line is called the *constraint line*. Although two slices are good enough to determine the *constraint line*, we use four to reduce the chance of alignment in a digitized process.

Besides the slope, the topology of *paths* in a *slice* also gives important information. In fig. 4-6 (a), the line segment to which the edgel **P** initially belongs, will occlude some other line segment x_o unit length away from **P** along the direction θ at time t_j. The same message is carried in figure 4-6 (b) except that the two lines are moving in opposite direction in (a) while both are moving in the same direction at different speed in (b). Figure 4-6 (c) and (d) show that **P** is on a line segment about to be occluded.

Figure 4-7 shows the cases of disocclusion, in which a new line segment shows up at the position x_d unit length away from **P** along the direction θ in the jth frame. The new line segment is slower than the current one in (a), faster in (b) and moving in a different direction in (c). Figure 4-8 shows some examples when a corner is encountered. Corners are worth noticing because they can give both velocity components of the motion.

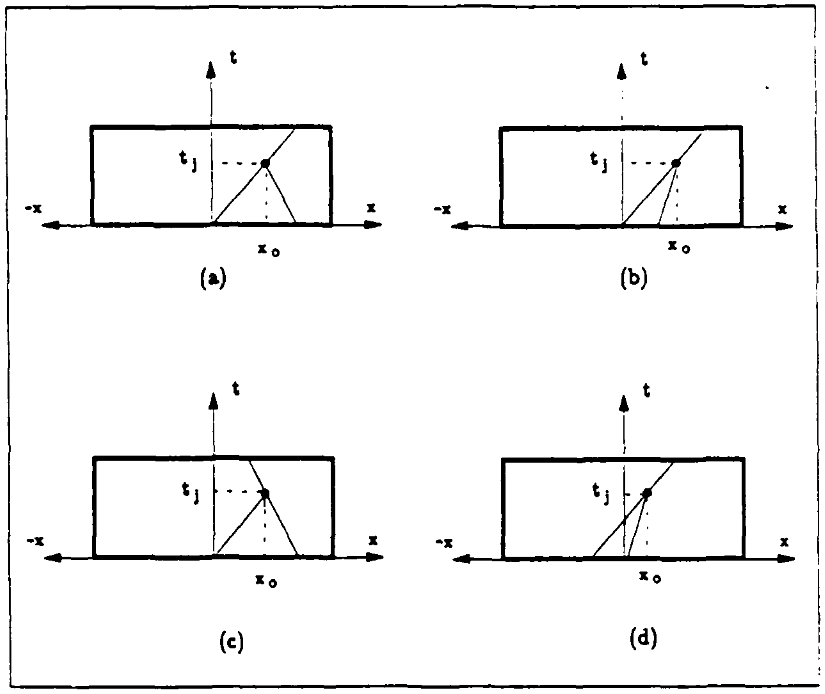


Figure 4-6: *Paths* with Occlusion

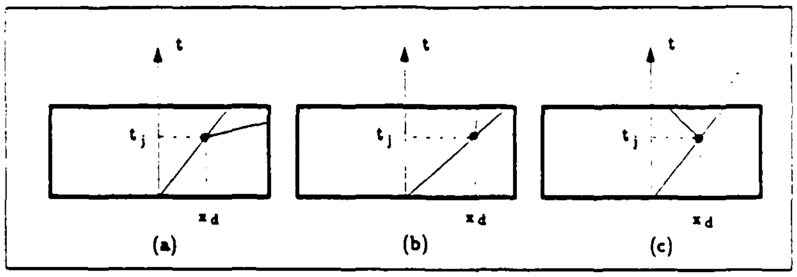


Figure 4-7: *Paths* with Disocclusion

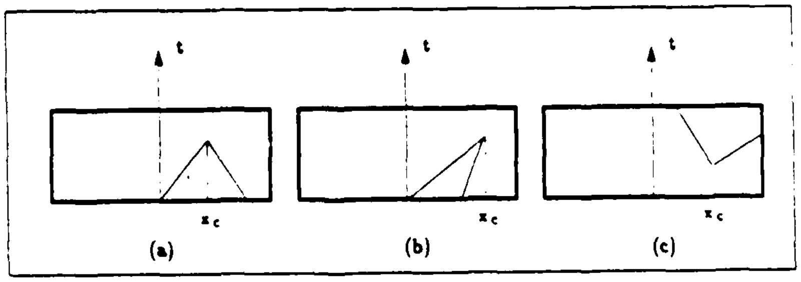


Figure 4-8: *Slice* with Corner

4.2.2 Segmentation based on motion only

Once we have the normal flows assigned to the edge points in a frame, the next step toward image understanding is the *interpretation* of the flow field, which can be subdivided into two stages, the first of these is to segment the edge points into contours and the second stage is to find the real velocities of the objects whose boundaries and surface markings give rise to those contours.

To segment edge points in an image frame without any *a priori* knowledge, problems may occur when there are more than one objects moving and occlusion and/or disocclusion take place. If one object is moving in front of another object then edge points on the boundaries of the rear surface will either be occluded or disoccluded during this movement, depending on whether the front object is moving to cover or uncover the object behind it.

The contours close to where the occlusion or disocclusion takes place will always form a three-way junction, where 2 branches belong to the front object while the third belongs to the rear one. One image frame along is not enough to tell which two branches go together. When we process the slices as mentioned in the previous section, the particular Y or λ shape paths will be noticed. Therefore, we can predict *where* and *when* the occlusions or disocclusions will happen, and send messages to the image frames to mark the places to watch out for occlusion or disocclusion.

Each frame will receive several messages from an earlier image frame if there exists occlusion or disocclusion in the frame. Besides the location, the messages also give the *dominant velocity* within the spots of occlusion or disocclusion. The *dominant velocities* is the velocity of the motion of the front object along the inclination of the slice, i.e. the slope of the crossing path which terminates the other. Now the segmentation becomes easier because the ambiguity of the three-way junctions is resolved. Whenever we trace a contour and get into an occlusion or disocclusion spot, we can use the similarities among the three branches to the *dominant velocities* to determine whether to break or extend the contour. Our segmentation program tries to generate contours as long as possible.

4.3 Computing the Correct Velocity Field along a Contour

The segmented contours are associated with the normal flow estimates of each point. An additional constraint is used for the integration of local motion measurement to compute the two-dimensional velocity field: the constraint is a smoothness constraint to minimize the variation of the velocity measurement along the contours, because the velocity field across a physical surface is generally expected to be smooth. The velocity field of least variation is in general not the physically correct one, however it is often qualitatively

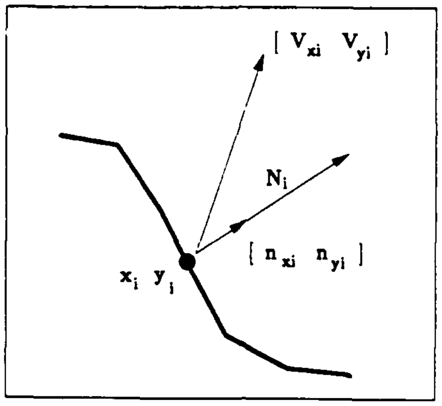


Figure 4-9: Illustration of the notations

similar to the true velocity field. When the two velocity fields differ significantly, it appears that the smoothest velocity field may be more consistent with human motion perception [11]. The particular measure of variation we choose is $\int_C \left|\frac{\partial \vec{V}}{\partial s}\right|^2 ds$: the integral of the square of velocity change along the contour.

If there exists at least two edge points at which the local orientation of the contour is different, then there exists an unique velocity field that satisfies the known normal velocities and minimize $\int_C \left|\frac{\partial \vec{V}}{\partial s}\right|^2 ds$. Since we have only discrete points, the first in the design of the algorithm is to convert the continuous formulation into a discrete one.

Assume that the contour has n edge points on it, { (x_0, y_0), (x_1, y_1), ..., (x_{n-1}, y_{n-1}) }. For each edge point, (x_i, y_i), see fig. 4-9, we have an estimate of the normal velocity represented by the magnitude of the normal velocity, N_i, and the direction normal, $[n_{xi}, n_{yi}]$, perpendicular to the contour. We want to find a list of velocities, { (Vx_0, Vy_0), (Vx_1, Vy_1), ..., (Vx_{n-1}, Vy_{n-1}) } which minimize the variation,

$$\sum_{i=1}^{n-1} \left[(Vx_i - Vx_{i-1})^2 + (Vy_i - Vy_{i-1})^2\right] \qquad (2)$$

and satisfies the constraint that the component of the velocity in the normal direction equals the estimated normal velocity,

$$Vx_i \cdot n_{xi} + Vy_i \cdot n_{yi} = N_i, \quad \text{i = 0 ... n-1} \qquad (3)$$

To loosen the constraints, equation (3) does not have to be exactly satisfied. Therefore the energy function, ϕ, is defined as a linear combination of the above two equations, which is

$$\begin{aligned}\phi = \sum_{i=1}^{n-1} \left[(Vx_i - Vx_{i-1})^2 + (Vy_i - Vy_{i-1})^2\right] \\ +\gamma \cdot \sum_{i=0}^{n-1} [Vx_i \cdot n_{xi} + Vy_i \cdot n_{yi} - N_i]^2\end{aligned} \tag{4}$$

To find out the set of velocities $\{ (Vx_0, Vy_0), (Vx_1, Vy_1), ..., (Vx_{n-1}, Vy_{n-1}) \}$ which minimizes the energy function in equation (4), one can take partial derivatives of ϕ with respect to Vx_i and Vy_i, where

$$\begin{aligned}\frac{\partial \phi}{\partial Vx_i} &= 0, \text{ and} \\ \frac{\partial \phi}{\partial Vy_i} &= 0, \text{ for i = 0 ... n-1}\end{aligned} \tag{5}$$

¿From equation 5 we have $2n$ linear equations for $2n$ unknowns. We can use any method to solve the linear system as long as not all the edge points are on a straight line. The method we choose is a *conjugate gradient algorithm*, which finds a solution in $2n$ iterations with the initial guess $Vx_i = N_i n_{xi}$ and $Vy_i = N_i n_{yi}$.

4.3.1 Higher level motion processing

After the segmentation and variation minimization, we have a set of contours associated with velocity estimates along them to represent the optical flow field in the dynamic scene. A great deal of information could be picked up from the flow field even without invocation of high level processing like object recognition. For example, the flow field is rich enough to support the inference of collision when a robot is moving in an unknown place, or to locate the *focus of expansion* for navigation.

The contours also outline the surface patches. With the velocities of the surfaces and their spatial relations in the two-dimensional scene, their three-dimensional structures and the three-dimensional motion may be determined.

In particular, it should follow that, away from the boundaries, adjacent pixels should have similar motion, pixels corresponding to the same physical location should have similar intensities, and the resulting path should be smooth [23]. Therefore the velocities assigned to the contours can be propagated into the *interior* of the surface, using Nagel's formulation [19] for instance, to generate a dense velocity field. We still have to make sure that the contour segments correspond to correct object boundaries. Otherwise the

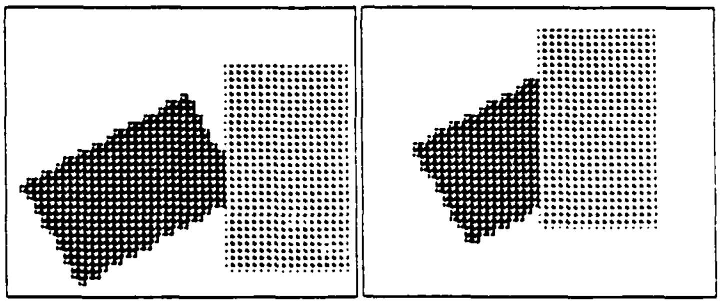


Figure 4-10: Synthetic Sequence of Two Rectangles

velocity fields might *bleed* into other irrelevant regions, which happens often if occlusion or disocclusion are not handled correctly.

4.4 Results

We have generated a sequence of synthetic images for the purpose of testing and illustration, in which there are two rectangles, see fig. 4-10. The longer axis of the leftmost rectangle makes a 30° with the x axis. It is moving in the north east direction half a pixel per frame, while the right rectangle is moving in the north west direction with the same speed. The first and twelfth frames are shown in fig. 4-10 (a) and (b). A typical example of the slice analysis is shown in fig. 4-11, in which an edge point on the lower left boundary of the left rectangle in the fourth frame is processed. The four slices are in the upper right pane while the right hand side shows the result of edge detection on the slices. The lower right pane shows the velocity points in the velocity space and the *constraint line* fitted to them, and the normal velocity assigned to the edge point. Figure 4-12 shows the normal flow by collecting all the estimates of normal velocity for edge points in the fourth frame. Figure 4-13 shows the result after segmentation and variation minimization. The fourth frame got several messages passed from previous frames and located the places of occlusion so that physically related edge points are grouped together. In this example, there are only two contours and the variation minimization algorithm is applied to both of them separately with $\gamma = 0.01$. The constructed velocity field is very close to the real value both quantitatively and qualitatively.

The next example is a real image sequences from SRI[1], in which the camera is moving forward in a lab. We only use a 32 by 32 window containing some palm leaves, since a small window is already good enough to demonstrate this local process. The first frame of the sequence with the window boundary highlighted is shown in fig. 4-14. The final

[1]Courtesy of Dr. Baker and Dr. Bolles

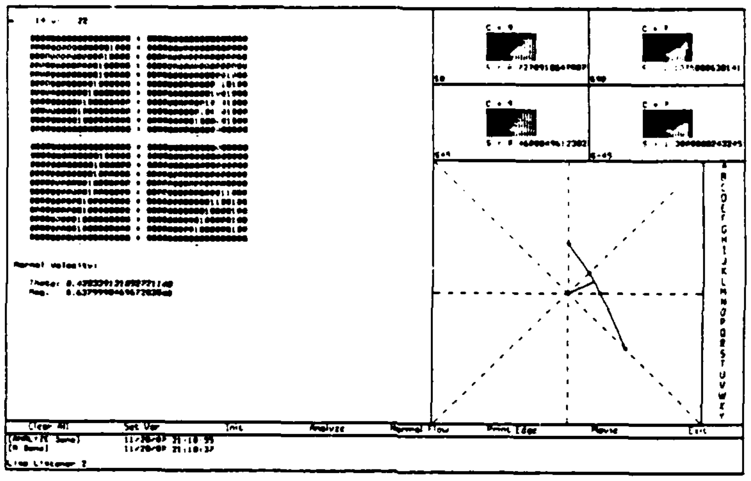


Figure 4-11: Slice Analysis

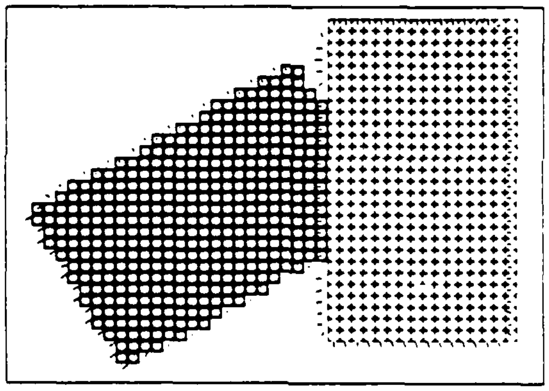


Figure 4-12: Normal Flow Field of the Fourth Frame

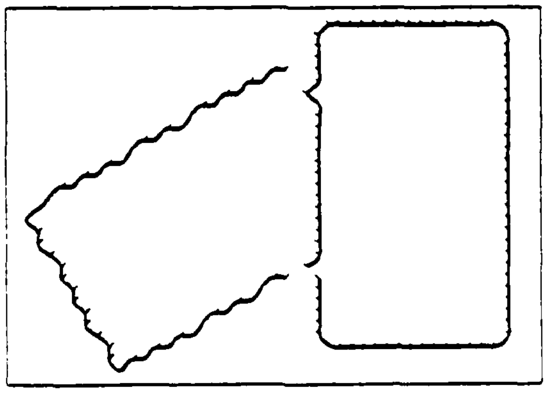


Figure 4-13: Segmented Contours and Velocity Field with $\gamma = 0.01$

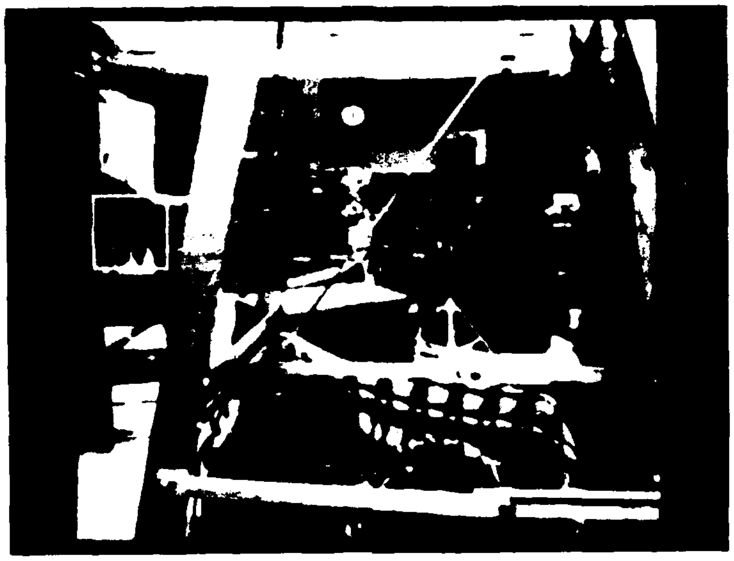


Figure 4-14: SRI Sequence: Zoom

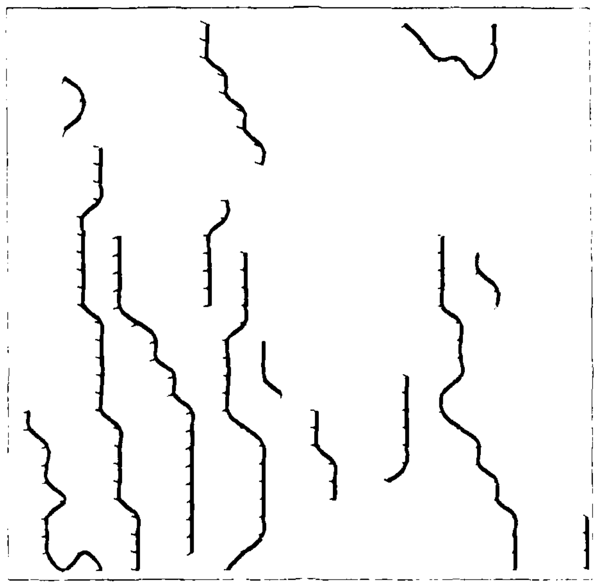


Figure 4-15: Segmented Contours and Velocity Field with $\gamma = 0.01$

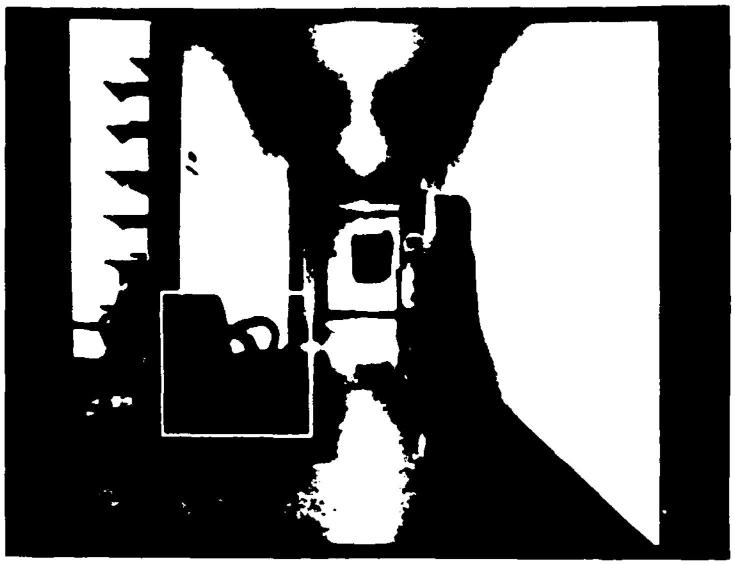


Figure 4-16: SRI Sequence: Hallway

result of segmentation and variation minimization is shown in fig. 4-15. Since the camera is pointing at some point to the right of the window while it moves, the sequence looks like the leaves are moving towards the viewer and passing by his left hand side with a little expansion. The last example is an image sequence taken by a robot while it is moving down a hallway. We choose a 64 by 64 window of the scene containing a chair against the wall, see fig. 4-16. The segmented contours and variation minimized velocity flow is shown in figure 4-17.

4.5 Conclusion

We have presented a spatio-temporal approach to solve the early processing problem of motion analysis, which can handle scenes of multiple moving objects with occlusion and/or disocclusion. The characteristics and advantages of this method are as follows:

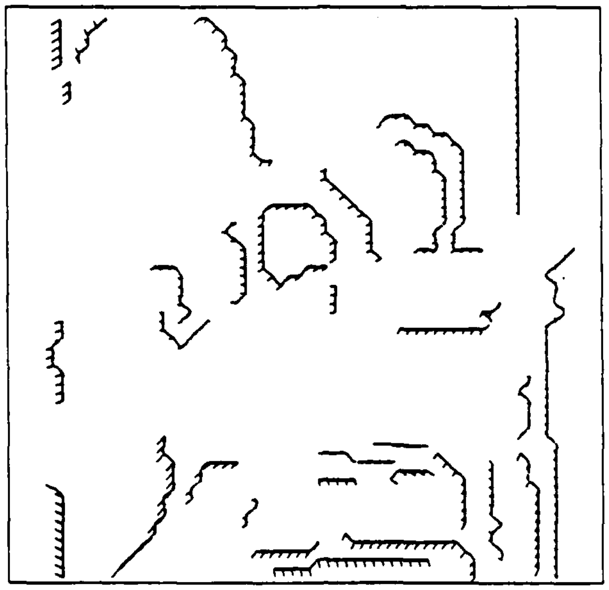


Figure 4-17: Segmented Contours and Velocity Field with $\gamma = 0.01$

1. *Parallel Processing in the Spatial Domain*
   The slice analyses of the edge points are independent so they can proceed in parallel. Assuming that there is a 2-D SIMD array as large as the image with each entry of the array an identical processing element, each processing element has a memory **L** words long and is able to talk to its neighbors **W** steps away in eight directions. All the processing elements can construct their slices at the same time via local communication and analyze the slices concurrently. This property promises a hardware implementation with identical units performing the same operations in parallel.

2. *Pipeline Processing in the Time Dimension*
   The slice analysis only takes **L** frames. Each time we pump in a new image frame, it is distributed over the entire array and each processing element simply fetches the corresponding pixel and gets rid of the oldest one in its local memory. While the array is working on segmentation and variation minimization of the current frame, the slice analysis can be invoked to work on the next frame.

3. *Symbolic Scene Description*
   The segmentation of moving objects are generated frame by frame, in which the contours provide the skeleton of the scene while the velocity field gives their relations over frames. Therefore, our process extracts suitable information for higher level interpretation process.

4. *Occlusion and Disocclution*
   Our method is able to identify occlusion and/or disocclusion, which is very important to circumvent ambiguity.

5. *Incremental Process*
   Our method can be extended such that after having processed a few frames, the system is able to predict the path in a slice. The predicted value can either reduce computation time or lead to better estimation.

Finally, with those benefits, one more point the authors would like to state is the liberal assumptions of this work. The assumptions are few and yet need to be loosely satisfied. Although the high sampling rate assumption allows us to approximate any short trajectory of movement by translation and approximate any portion of the object boundary by piecewise straight line, we want to extend this work to handle rotation and motion at varying speeds. The only change is to allow curved paths in a slice instead of purely straight lines. The only ambiguity is that the paths for motion at changing speed and for curved object boundary are both curved. More work is also required to study the effects of quantization and noise sensitivity in a larger number of real image sequences.

References

[1] P. Anandan, " A Unified Perspective on Computational Techniques for Measurement of Visual Motion," *Proceedings of Image Understanding Workshop*, Vol. 2, February 1987, pp. 719-732.

[2] M. Asada, M. Yachida, and S. Tsuji, "Three dimensional Motion Interpretation for the Sequences of Line Drawings," *Proceedings of the International Conference on Pattern Recognition*, 1980, pp. 1266-1273.

[3] S. T. Barnard and W. B. Thompson, "Disparity Analysis of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 4, July 1980, pp. 333-340.

[4] R. C. Bolles and H. H. Baker, "Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences," *Proceedings of the Third Workshop on Computer Vision: Representation and Control*, Bellaire, Michigan, October 1985, pp. 168-178.

[5] J. Canny, " A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, November 1986, pp. 679-698.

[6] L. S. Davis, Z. Wu, and H. Sun, "Contour-Based Motion estimation," *Computer Vision, Graphics and Image Processing*, Vol. 23, 1983, pp. 313-326.

[7] L. Dreschler and H.-H. Nagel, "Volumetric Model and 3-D Trajectory of a Moving Car Derived from Monocular TV-frame Sequence of a Street Scene," *International Joint Conference on Artificial Intelligence*, Vancouver, Canada, August 1981.

[8] C. L. Fennema and W. B. Thompson, "Velocity Determination in Scenes Containing Several Moving Objects," *Computer Graphics and Image Processing*, Vol. 9, April 1979, pp. 301-315.

[9] D. B. Genery, "Object Detection and Measurement Using Stereo Vision," *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, August 1979.

[10] F. Glazer, G. Reynolds and P. Anandan, "Scene Matching by Hierarchical Correlation," *Proceedings of Computer Vision and Pattern Recognition*, June 1983, pp.432-441.

[11] E. C. Hildreth, "The Computation of the Velocity Field," Technical report 734, Massachusetts Institute of Technology, September 1983.

[12] C. J. Jacobus, R. T. Chien and J. M. Selander, "Motion Detection and Analysis by Matching Graphs of Intermediate Level Primitives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 6, November 1980, pp. 495-510.

[13] R. Jain and H.-H. Nagel, "On the Analysis of Accumulative Difference Pictures from Image Sequences of Real World scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 1, No. 2, April 1979, pp. 206-214.

[14] R. Jain, "Extraction of Motion Information from Peripheral Processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 3, September 1981, pp. 489-503.

[15] S. Y. Kung, "On Supercomputing with Systolic/wavefront Array Processors," *IEEE Proceedings*, Vol. 72, No. 7, July 1984, pp. 867-884.

[16] D. H. Marimont, "Projective Duality and the Analysis of Image Sequences," *Proceeding of IEEE Workshop on Motion: Representation and Analysis*, May, 1986, pp. 7-14.

[17] G. Medioni and R. Nevatia, "Segment-Based Stereo Matching," *Computer Vision, Graphics, and Image Processing*, Vol. 31, 1985, pp. 2-18.

[18] H. P. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*, Ph. D. dissertation, Stanford University, 1980, also available as Carnegie-Mellon University RI-RT-3, Robotics Institute.

[19] H.-H. Nagel and W. Enkelmann, "An investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields from Image Sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 5, September 1986, pp. 565-593.

[20] J. M. Prager, "Segmentation of Static and Dynamic Scenes," Technical Report TR 79-7, Dept. of Computer and Information Science, University of Massachusetts, May 1979.

[21] K. Price and R. Reddy, "Matching Segments of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 1, No. 1, January 1979, pp. 110-116.

[22] V. S. Ramachandran and S. M. Anstis, "The Perception of Apparent Motion," *Scientific American*, June, 1986, pp. 102-109.

[23] I. K. Sethi and R. Jain, "Finding Trajectories of Feature Points in a Monocular Image Sequence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 1, Jan. 1987, pp. 56-73.

[24] M. A. Shah and R. Jain, "Detecting Time Varying Corners," *Computer Vision, Graphics, and Image Processing*, Vol. 28, 1984, pp. 345-355

[25] W. E. Snyder, "Computer Analysis of Time Varying Images," *Computer*, Vol. 14, No. 8, August 1981, pp. 7-9.

[26] J. Tanner and C. Mead, "An Integrated Analog Optical Motion sensor," *IEEE ASSP Society Workshop on VLSI Signal Processing, II* November, 1986, pp. 59-76.

[27] T. D. Williams, "Depth of Camera Motion in a Real World Scene," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 6, November 1980, pp. 511-516.

5 INTEGRATION EFFORT IN KNOWLEDGE-BASED VISION TECHNIQUES FOR THE AUTONOMOUS LAND VEHICLE PROGRAM

Over the past several years the USC Computer Vision group has developed a number of component programs that can be applied to motion analysis in the Autonomous Land Vehicle (ALV). Thus, we have a number of separate programs (or collection of programs) developed by different people for different computer vision tasks [1], [2], [3], [Section 2 of this report] with no strict requirements imposed on the developers as to what input, output and program parameters should be used. We will use the word *module* to refer to a collection of programs that are solving a particular task from the computer vision domain (for example, a collection of programs that find depth of environmental points using a pair of stereo images). Our current task is to construct a control structure that will use these different modules and enable them to cooperate in visual guidance of an ALV using general motion analyzing techniques.

We consider the integration to be an important effort for several reasons. Different feature extraction or matching techniques may work best in specific circumstances, thus a variety of modules for similar operations are necessary. Additionally, it is too costly and time-consuming to reprogram current modules into a coherent and unified computer vision program. Even if we would succeed in this effort, we would lose the generality of using the same basic modules for multiple applications. By using a variety of modules for similar operations (e.g., feature matching) we will develop techniques that can more easily accept other, newer, modules for the same or related processing steps. Therefore, we prefer using the current modules, at the expense of designing a control structure for them. In order to create the task configurations we must understand how the modules interact and the type of interfaces needed between modules.

The problem we face is not the problem of the top-down design which had divided the task into subtasks that will be later linked together. There, predetermined data structures enable easy module integration. We did not influence the design and the interaction of modules, although the modules may had been developed for similar domains end therefore could share similar input data.

There are several important issues we had to address in the integration effort. We chose not to create a completely general control structure and interface (such as a blackboard) because of the desire to quickly incorporate current results in component development. Here we present these issues in general terms and will later give some examples and initial results.

1. If we know how to combine different modules (in principle) what is it that we have

to do (in practice) to make the combination of the modules work together? We call this an *interface* problem. The question is then to design interfaces in a manner that hides details of implementation of one module from another module without loosing the capabilities already present in either module.

2. How are we going to judge the *performance* of the combined modules once they are "sewn together?" If a human operator must assess the performance of the modules on a particular subtask, we will be not able to combine several modules to work automatically on a more complex task.

3. Can the system suggest (and eventually generate automatically) a *configuration* of the modules that will be the best for a given task? How can we incorporate the knowledge that people use when choosing a set of programs to perform some visual perception task?

4. Somewhat related to the previous two issues is: should we strive for a *static* or *dynamic integration*? In a static integration two modules are "hardwired" feeding input or output to each other (if a feedback is used) but essentially they are forced to work together independently of the input domain. Dynamic integration links modules at the run-time depending on the domain, module performances and the task in question. Dynamic integration is a much more complex problem and we are not in the position yet to attack this problem.

5.1 Control Structure

In this section we outline the major design decisions that are made in the integration effort of different motion modules. In the next section we will present our results and the current state of the integrated software.

In the design of an integrated software for a particular task, the first step is to define module's input, output, its preconditions (range of parameters), purpose, efficiency, end expected quality of the results. The next step is the design of the control program, that will synchronize the work of motion modules. We have used a similar design strategy for our motion integration package as those found in design of software for automatic programming, in particular those found in work by Kant [4]. Although the task system is designed for the domain of motion analysis of the ALV, most of the decisions are equally applicable to a general purpose vision system.

5.1.1 Design decisions

The system has four major components:

- The set of routines that specify modules and create a module configuration needed for particular task (task definitions).
- The set of routines that schedule and execute a particular task (task execution).
- The set of routines that create history of data and control flow. The user can then examine intermediate results, and rerun tasks perhaps using different modules or data.
- A user-friendly interface that allows the easy modification of input and output parameters and the easy design of new task configurations. It also helps in displaying the history of the run using images and data tables.

For each module we define the module components, function and required input and output data. We separate functionality of the module from its domain and range, so that we can create separate data and control flows [5]. This decision helps the user create a task configuration and build a control structure on top of the latter. Knowledge-based scheduling and execution require the separation of data and control flow for the same reason.

Task configurations are represented as graphs in which modules are nodes and paths are data and/or control flows. Each node (module) can have several input and/or output data ports because the type of the data required by modules greatly varies. In addition to the input and output data required for the module, there are also parameters for internal graphic displays and debugging information from the module. One of the important module characteristics is that they can be implemented in different programming languages (like C and Lisp in our case) and the data ports provide a convenient interface between these modules.

Modules can be configured using different control structures (loops, sequences, concurrent execution, conditional constructs, etc.). This means that we can use data feedback between modules, use several machines to run modules concurrently if needed, and make choices about module execution depending on the data they use. The design allows both forward-chaining and goal-directed reasoning which is needed in a more sophisticated task scheduling and execution environment.

The task history is a very helpful tool for rerunning the same set of module executions, examining the data (images, parameters) at each stage of execution, perhaps selecting a new set of parameters for a new run, and serving as a quick demonstration facility.

Task configuration, execution and history are accessible to the user through the graphic interface. A powerful graphic editor is used to help the user to: compose and decompose task configurations; enable task interruption or execution; examine, edit or save data used or produced at different stages of task execution; edit the global knowledge base; and enable task rerun. The graphics editor also has tools for data smoothing and data routing and conversion to and from different computational machines.

All the above program decisions are made in order to allow an interactive, user-friendly, problem-solving motion package that can be later used in a semi-automatic or automatic way to detect the environmental changes from images. Modularity of the design enables easy incremental addition or change of modules or data, and a greater flexibility and efficiency of the whole knowledge-based motion system.

5.1.2 Interface problem

In this subsection we discuss issues in creating interfaces between different modules. We initially have developed a set of specific module interfaces rather than a single data transfer mechanism so that we can concentrate on computer vision rather than general system building. The other reason is that we currently have only a few modules for each subproblem and the design of interfaces between each pair of modules is not a costly design decision.

The long term effort requires that the interfaces between the two modules are general enough to handle not only the particular pair of modules, but a pair of classes of modules. A class of modules has elements that solve one specific problem of the vision, for example, all the modules that perform straight line extraction will be one class (say class A), and all the modules that find line correspondences will be in another class (say class B). The interface between any module from class A and any module from class B will be the same. The reason for such a design is that we would like to handle lines as semantic entities and not be concerned with detailed representation of the line.

The other important issue is a need to design these interfaces for possible use in a feedback loop. In these situations the output of the second module might be used to improve the performance of the modules that provided its input.

Interfaces hide details between the different requirements of different modules. In the example that we present in the next section, a motion estimation module requires the position of a region in several frames. On the other hand, a region matching module returns corresponding regions between two frames. The interface between matching and motion estimation modules accumulates the pairwise matches until enough are found for the motion estimation module. In other situations, the interfaces hide the details of data representation for different modules because we are only concerned with semantic notion of features and not its representation.

Sometimes an interface must account for missing data, and sometimes it should discard data that are not considered to be essential. We plan to equip the input and output data structures with procedures that will signal the absence of necessary data, so that the missing data could be recovered by calling some other module, or the user.

5.2 Results

Our initial implementation provides a working prototype and a baseline system for testing of the integration framework. It is implemented in an object-oriented language (flavors). In this implementation, the initial control program that drives different modules is "hard-wired," .hus avoiding several important issues that usually appear in automatic programming. However, we still had to solve the interface problem. We have also implemented the most important parts of the user-interface.

As the initial step to integration, and to provide a convenient method to more easily test the motion estimation system, we have combined modules for feature extraction using the region segmentation techniques [6], feature matching using our region based matching system [2], three-dimensional motion estimation [3], and feedback of the image location prediction to the matching programs. These programs were written by different authors, without considering the need to integrate these specific programs into one system, thus some of the effort is required to transform the data produced by one system into data expected by the next. For example, the matching system provides a symbolic description of the two input images with links between them and the motion estimation program only requires a list of point correspondences for several frames. The list of points can be derived from the matching output.

This initial integrated system demonstrates the ability to combine different subsystems into one unified system. This prototype system has the following tasks (see Figure 5-1 for a description of the current system):

- **Image input:** Read the image sequence.
- **Image segmentation:** With large images and for time considerations, a subimage is segmented into regions by the histogram based segmentation program. These regions may, or may not, correspond to actual real-world objects, but are assumed to be single objects for the purpose of motion segmentation. All the images in the sequence are initially segmented. Features of individual regions and relations between regions are also computed. One of the features, the center of mass, is used later in the motion estimation module. This forms the symbolic description of the image.
- **Match the first image description to the second:** Initially, there is no information to guide the match, so the first few matching steps must use the general

techniques with features such as intensity, size, shape, adjacencies, relative positions, etc. This produces a set of corresponding regions where a region in the first view is paired with a region in the second view.

- **Match the second image description to the third:** This step is the same as the previous one where general features must be used. At this point the translation estimation module used for generating predictions of future image plane locations for those regions that are tracked from image 1 to 2 to 3 (i.e., region X in image 1 is matched to region Y in image 2 which is then matched to region Z in image 3). The general motion estimation system requires one point in five consecutive frames, but three dimensional translation can be computed using only one point in three frames. Thus, the matching through three frames allows the prediction of the region location in the forth and future frames.

- **Continue the matching process for descriptions of image N to image N+1:** Since a motion estimate has been computed for some of the regions, the predicted position of the region in the next image can be used as a feature (the position) in the matching process. This allows greater motions to be easily handled by the later matches. The motion estimation programs (general estimation for 5 or more frames in a sequence and translation estimation for 3 or 4) are applied on each sequence of matching regions.

The motion estimation results are displayed in several forms at each stage, including the trajectory mapped back onto the image plane (using perspective projection), an orthographic projection of the trajectory viewed from the top, and another viewed from the side. These three displays are given in Figure 2, with the perspective view showing the motion of the four regions (grill (3,6), bumper (2,7), front shadow (1,4) and side shadow (4)) drawn for the six frames in the sequence and drawn on the next-to-last (fifth) frame. The computed motion projections for all the regions, except the side shadow, are shown for frames 1 through 5 (labeled 1, 2 and 3) and for frames 2 through 6 (labeled 4, 5, 6 and 7). The two orthographic views show that the motion is completely in the Z and X directions (see the side view motion where Y is almost constant for each region) and shows the trajectories of the regions in the correct relative positions. These three-dimensional trajectories are scaled to the dimensions of the focal plane of the camera since absolute scale can not be derived. The positions are also adjusted for the computed relative depth of the points as shown by the fact that the beginning locations for points 4-6 are closer to the camera (i.e., Z is smaller) than the beginning locations for points 1-3.

This version of the program was intended only as a test of the current component interfacing and to provide an outline for the future system.

5.3 Future Work

As we have seen in the previous section the initial motion integration package performs region segmentation, evaluates region correspondences not for a single pair but for many pairs of frames. Then an estimation motion module is called that determines the motion parameters of the ALV. Major areas for future work include using more feedback from motion estimation to matching and using feedback from both motion and matching to segmentation.

We also plan to add in the motion detection system another subsystem that uses a Hough-transform based module to detect preliminary line correspondences. The later will provide input to a module for more precise line-correspondences (these two might be connected via a feedback loop). The results of these two modules are to be fed into a third module that uses line-correspondences in several frames to detect motion of objects in the scene. The results on this subsystem will be reported later.

We will use the contour based matching approach [Section 2 of this report] for direct input to the motion estimation programs and plan to combine it with the region based matching system. This will allow the detailed matching results using contours to be computed when the motion between frames is large.

These three examples demonstrate that we have different input situations in mind (some images suitable for region matching, some for straight, some for curved line matching), and that each group of modules will be used depending on the input data. That is an example of what is needed in a more general purpose vision guidance system.

References

[1] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, 1980.

[2] O.D. Faugeras and K. Price. Symbolic description of aerial images using stochastic labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(6):633–642, November 1981.

[3] H. Shariat and K. Price. Results of motion estimation with more than two frames. In *Proceedings of the DARPA Image Understanding Workshop*, pages 694–703, Los Angeles, California, 1987.

[4] E. Kant. Interactive problem solving with a task configuration and control system. Technical report, Schlumberger-Doll Research, December 1987. Technical Report.

[5] D.R. Barstow. Automatic programming for streams ii: Transformational implementation. Technical report, Schlumberger-Doll Research, August 1987. Technical Report.

[6] R. Ohlander, K. Price, and R. Reddy. Picture segmentation by a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

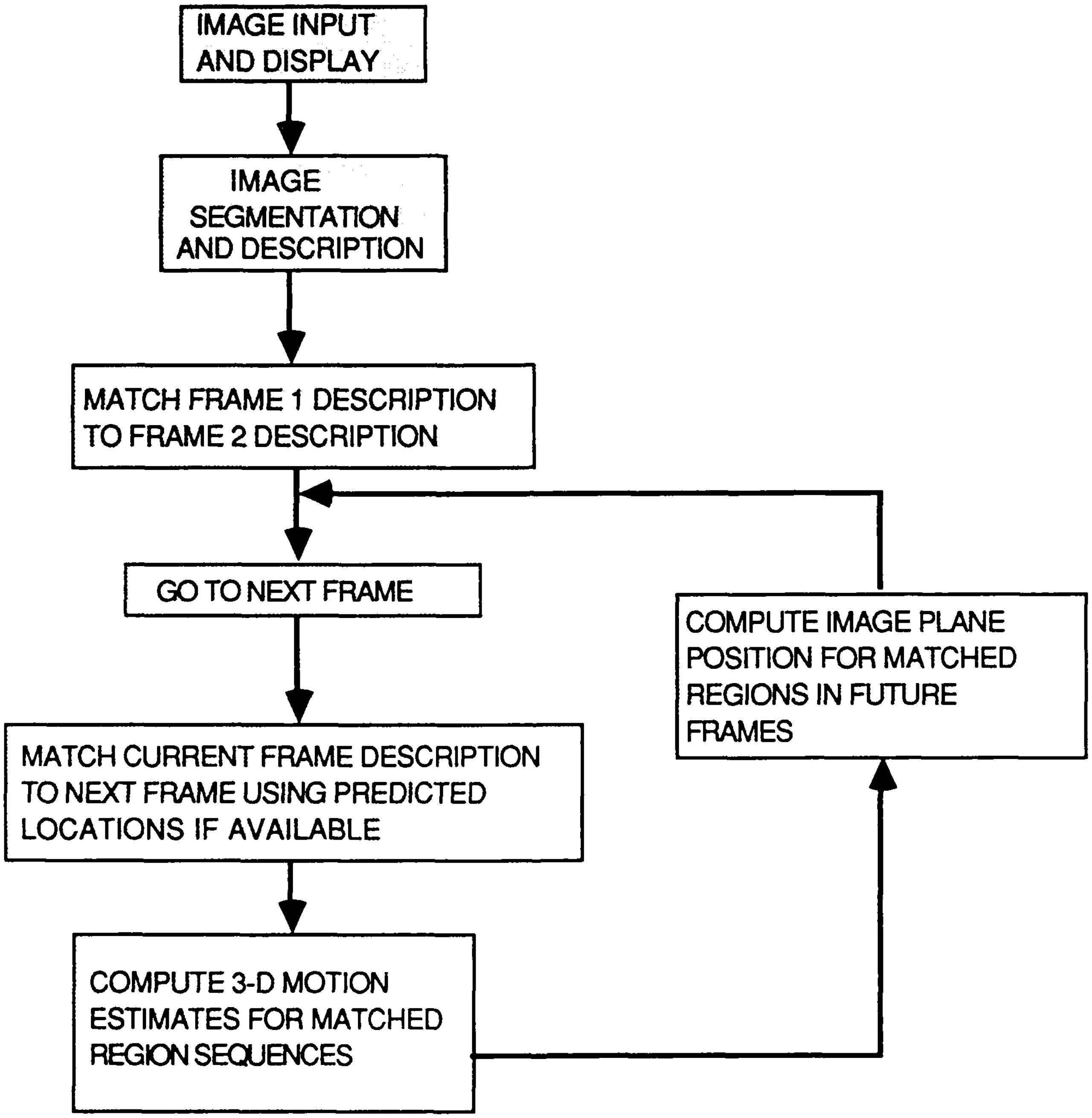


IMAGE INPUT AND DISPLAY
IMAGE SEGMENTATION AND DESCRIPTION
MATCH FRAME 1 DESCRIPTION TO FRAME 2 DESCRIPTION
GO TO NEXT FRAME
MATCH CURRENT FRAME DESCRIPTION TO NEXT FRAME USING PREDICTED LOCATIONS IF AVAILABLE
COMPUTE 3-D MOTION ESTIMATES FOR MATCHED REGION SEQUENCES
COMPUTE IMAGE PLANE POSITION FOR MATCHED REGIONS IN FUTURE FRAMES

Figure 5-1: Description of Current System

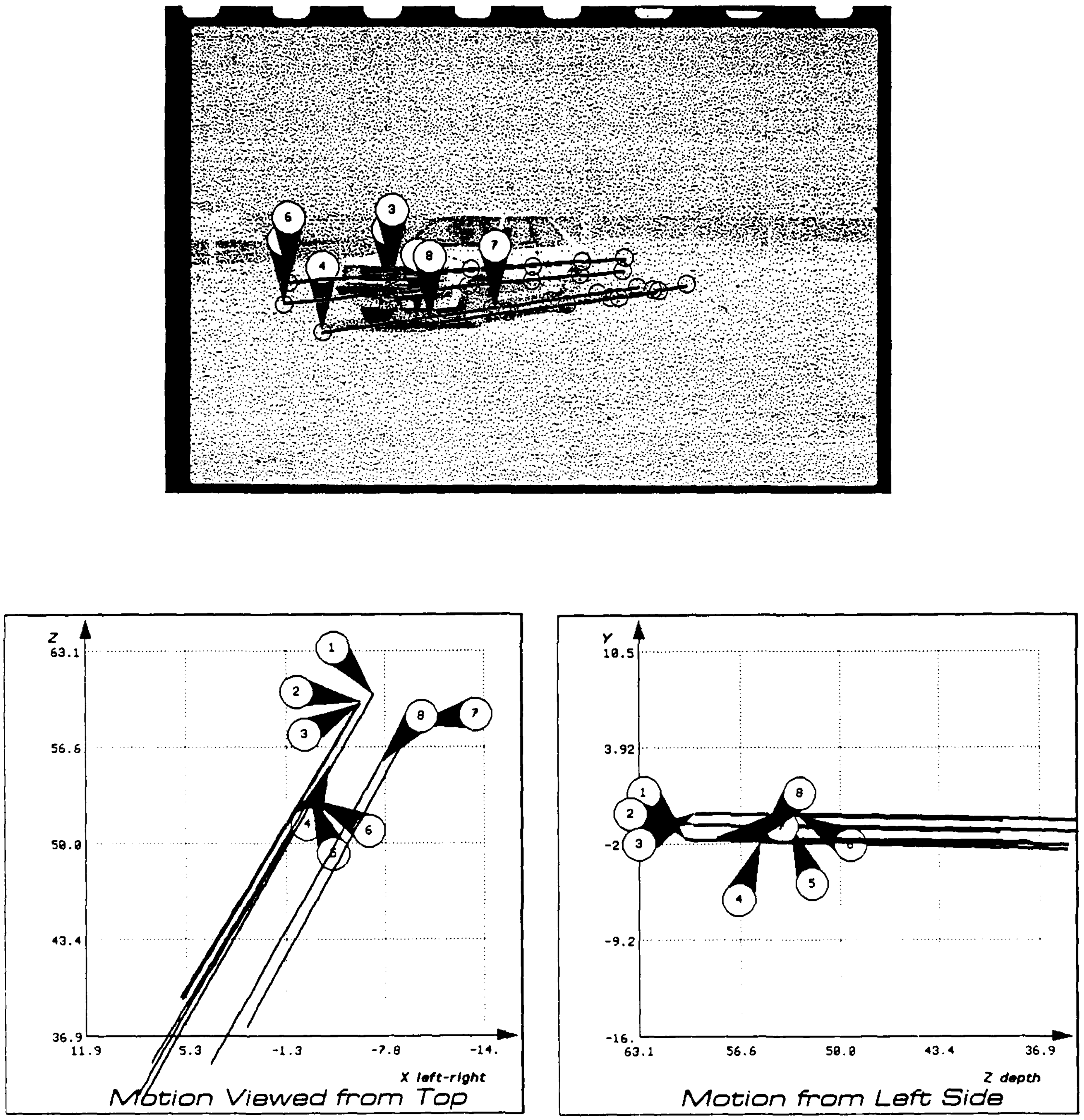


Figure 5-2: Results

6 FUTURE RESEARCH

For the next year we plan to continue the basic efforts described in this report. This will include the completion of the contour matching system and its integration into our basic motion analysis system. The other efforts are expected to continue, with only partial completion of each of the projects. In more detail, our expected work for the next year includes:

The incorporation of our basic motion analysis system into the CMU testbed facility and/or the Martin-Marietta ALV testbed. This is not an effort to tightly couple our motion analysis system with the ALV system, but an effort to show that it can operate in a more realistic environment. We will also continue the integration of the various feature extraction and matching subsystems with the motion estimation subsystem, by allowing for feedback of the estimated positions from the motion system to the matching programs. This also includes the effort to implement a more complete integrated motion analysis system that has the contour extraction and matching systems in addition to the region matching systems. This system should provide some depth and structure information in addition to the motion estimations.

With the increasing availability of motion sequence data, we can demonstrate the effectiveness of the matching and estimation subsystems on more sequences. This helps explore the limits of the algorithms and indicate where more efforts are needed to build a complete system.

The contour based matching system is nearing completion, but we will continue work to handle much larger differences between images and to track the matching points on the contour through several frames. The several frame matching is necessary to provide data to the motion estimation system, which requires matches through at least 3 frames (and 5 for accurate translation or general motions) for estimation.

This past year we did the theoretical development of the chronogeneous coordinate representation technique. We plan to begin the implementation of a motion estimation program using the chronogeneous coordinate representation. The development will build on our current motion estimation programs, using many common pieces and will enable the motion system to consider certain accelerations in addition to the other motions.